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, page 1
- Restrictions for Network-Based Recording, page 2
- Information About Network-Based Recording Using CUBE, page 3
- How to Configure Network-Based Recording, page 8
- Additional References for Network-Based Recording, page 27

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

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio-only Stream Forking of Video Call</td>
<td>Cisco IOS 15.4(3)M</td>
<td>The Audio-only Stream Forking of Video Call feature supports CUBE-based forking and recording of only audio calls in a call that includes both audio and video. The following commands were introduced: media-type audio.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.13S</td>
<td></td>
</tr>
<tr>
<td>Network-Based Recording of Video Calls Using CUBE</td>
<td>Cisco IOS 15.3(3)M</td>
<td>The Network-Based Recording of Video Calls using CUBE feature supports forking and recording of video calls.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.10S</td>
<td></td>
</tr>
<tr>
<td>Network-Based Recording of Audio Calls Using CUBE</td>
<td>Cisco IOS 15.2(1)T</td>
<td>The Network-Based Recording of Audio Calls using CUBE feature supports forking for RTP streams. The following commands were introduced or modified: media class, media profile recorder, media-recording, recorder parameter, recorder profile, show voip recmsp session.</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XE 3.8S</td>
<td></td>
</tr>
</tbody>
</table>

Restrictions for Network-Based Recording

- Network-based recording is not supported for the following calls:
  - Calls that do not use Session Initiation Protocol (SIP). Must be a SIP-to-SIP call flow
  - Flow-around calls
  - Session Description Protocol (SDP) pass-through calls
  - Real-time Transport Protocol (RTP) loopback calls
  - High-density transcoder calls
  - IPv6-to-IPv6 calls
  - IPv6-to-IPv4 calls with IPv4 endpoint.
  - Secure Real-time Transport Protocol (SRTP) passthrough calls
  - SRTP-RTP calls with forking for SRTP leg (forking is supported for the RTP leg)
  - Resource Reservation Protocol (RSVP)
  - Multicast music on hold (MOH)
• Any media service parameter change via Re-INVITE or UPDATE from Recording server is not supported
  Midcall renegotiation and supplementary services can be done through the primary call only.
• Media service parameter change via Re-INVITE or UPDATE message from the recording server is not supported
• Recording is not supported if CUBE is running a TCL IVR application.
• Media mixing on forked streams 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.

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 flow through CUBE.
4. CUBE sets up a new SIP session with MediaSense based on policy.
5. CUBE forks RTP media to MediaSense. For an audio call, audio is forked. For a video call, both audio and video are forked. For an audio-only configuration in a audio-video call, only audio is forked. There will be two or four m-lines to the recording server, based on the type of recording.

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

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

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

**Open Recording Architecture**

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

**Network Layer**

The ORA network layer is comprised of call control systems, media sources, and IP foundation components, such as routers and switches.

**Capture and Media Processing Layer**

The ORA capture and media processing layer includes core functions of ORA—terminating media streams, storage of media and metadata, and speech analytics that can provide real-time events for applications.

**Application Layer**

The ORA application layer supports in-call and post-call applications through open programming interfaces. In-call applications include applications that make real-time business decisions (for example, whether to record a particular call or not), control pause and resume from Interactive Voice Response (IVR) or agent desktop systems, and perform metadata tagging and encryption key exchange at the call setup.

Post-call applications include the following:

- Traditional compliance search, replay, and quality monitoring.
- Advanced capabilities, such as speech analytics, transcription, and phonetic search.
• Custom enterprise integration.
• Enterprise-wide policy management.

Media Forking Topologies

The following topologies support media forking:

Media Forking with Cisco UCM

The figure below illustrates media forking with Cisco Unified CallManager (Cisco UCM) topology. This topology supports replication of media packets to allow recording by the caller agent. It also enables CUBE to establish full-duplex communication with the recording server. In this topology, SIP recording trunk is enhanced to have additional call metadata.

Media Forking without Cisco UCM

The topology below shows media forking without the Cisco UCM topology. This topology supports static configuration on CUBE and the replication of media packets to allow recording caller-agent and full-duplex interactions at an IP call recording server.
SIP Recorder Interface

SIP is used as a protocol between CUBE and the MediaSense SIP server. Extensions are made to SIP to carry the recording session information needed for the recording server. This information carried in SIP sessions between the recording client and the recording server is called metadata.

Metadata

Metadata is the information that is passed by the recording client to the recording server in a SIP session. Metadata describes the communication session and its media streams.

Metadata is used by the recording server to:

• Identify participants of the call.
• Associate media streams with the participant information. Each participant can have one or more media streams, such as audio and video.
• Identify the participant change due to transfers during the call.

The recording server uses the metadata information along with other SIP message information, such as dialog ID and time and date header, to derive a unique key. The recording server uses this key to store media streams and associate the participant information with the media streams.
How to Configure Network-Based Recording

Configuring Network-Based Recording (with Media Profile Recorder)

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td><code>media profile recorder profile-tag</code></td>
<td>Configures the media profile recorder and enters media profile configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device(config)# media profile recorder 100</code></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>4</td>
<td><strong>media-type audio</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaprofile)# media-type audio</td>
</tr>
<tr>
<td>5</td>
<td><strong>media-recording dial-peer-tag [dial-peer-tag2...dial-peer-tag5]</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaprofile)# media-recording 8000 8001 8002</td>
</tr>
<tr>
<td>6</td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaprofile)# exit</td>
</tr>
<tr>
<td>7</td>
<td><strong>media class tag</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(config)# media class 100</td>
</tr>
<tr>
<td>8</td>
<td><strong>recorder profile tag</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaclass)# recorder profile 100</td>
</tr>
<tr>
<td>9</td>
<td><strong>exit</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(cfg-mediaclass)# exit</td>
</tr>
<tr>
<td>10</td>
<td><strong>dial-peer voice dummy-recorder-dial-peer-tag voip</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(config)# dial-peer voice 8000 voip</td>
</tr>
<tr>
<td>11</td>
<td><strong>media-class tag</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# media-class 100</td>
</tr>
<tr>
<td>12</td>
<td><strong>destination-pattern [+] string [T]</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>Device(config-dial-peer)# destination-pattern 595959</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Step 13</strong> session protocol sipv2</td>
<td>Configures the VoIP dial peer to use Session Initiation Protocol (SIP).</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session protocol sipv2</td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong> session target ipv4:[recording-server-destination-address</td>
<td>Specifies a network-specific address for a dial peer.</td>
</tr>
<tr>
<td></td>
<td>recording-server-dns]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session target ipv4:10.42.29.7</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong> session transport tcp</td>
<td>Configures a VoIP dial peer to use Transmission Control Protocol (TCP).</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# session transport tcp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</strong> end</td>
<td>Returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config-dial-peer)# end</td>
<td></td>
</tr>
</tbody>
</table>
## Configuring Network-Based Recording (without Media Profile Recorder)

### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
</tbody>
</table>
| **Example:** | `
Device> enable` |
| **Step 2** | Enters global configuration mode. |
| `configure terminal` | Enters global configuration mode. |
| **Example:** | `
Device# configure terminal` |
| **Step 3** | Configures the media class and enters media class configuration mode. |
| `media class tag` | Configures the media class and enters media class configuration mode. |
| **Example:** | `
Device(config)# media class 100` |
| **Step 4** | Enters media class recorder parameter configuration mode to enable you to configure recorder-specific parameters. |
| `recorder parameter` | Enters media class recorder parameter configuration mode to enable you to configure recorder-specific parameters. |
| **Example:** | `
Device(cfg-mediaclass)# recorder parameter` |
### Configuring Network-Based Recording (without Media Profile Recorder)

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ipv4: destination address --IP address of the dial peer, in this format: xxx.xxx.xxx.xxx</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**
```
Device(config-dial-peer)# session target ipv4:10.42.29.7
```

### Step 14

**session transport tcp**

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

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

### Step 15

**end**

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

**Purpose:** Returns to privileged EXEC mode.

---

### Verifying the Network-Based Recording Using CUBE

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

### SUMMARY STEPS

1. enable
2. show voip rtp connections
3. show voip recmsp session
4. show voip recmsp session detail call-id call-id
5. show voip rtp forking
6. show call active voice compact
7. show call active video compact
8. show sip-ua calls
9. show call active video brief
10. debug ccsip messages (for audio calls)
11. debug ccsip messages (for video calls)
12. debug ccsip messages (for audio-only recording in a call with both audio and video)
13. Enter one of the following:
   - debug ccsip all
   - debug voip recmsp all
   - debug voip ccapi all
   - debug voip fpi all (for ASR devices only)
DETAILED STEPS

Step 1  enable
Enables privileged EXEC mode.

Example:
Device> enable

Step 2  show voip rtp connections
Displays Real-Time Transport Protocol (RTP) connections. Two extra connections are displayed for forked legs.

Example:
Device# show voip rtp connections
VoIP RTP Port Usage Information:
Max Ports Available: 8091, Ports Reserved: 101, Ports in Use: 8
Port range not configured, Min: 16384, Max: 32767

<table>
<thead>
<tr>
<th>Media-Address Range Available</th>
<th>Ports Reserved</th>
<th>Ports In-use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ports</td>
<td>8091</td>
<td>101</td>
</tr>
</tbody>
</table>

VoIP RTP active connections :
No. CallId dstCallId LocalRTP RmtRTP LocalIP          RemoteIP
1   1   2   16384 20918 10.104.45.191              10.104.8.94
2   2   1   16386 17412 10.104.45.191              10.104.8.98
3   3   4   16388 29652 10.104.45.191              10.104.8.98
4   4   3   16390 20036 10.104.45.191              10.104.8.94
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

Anchor Leg 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

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

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

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

<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream State</td>
<td>Displays the state of the call. This can be ACTIVE or HOLD.</td>
</tr>
</tbody>
</table>
### Output Field | Description
---|---
Msp Call-Id | Displays an internal Media service provider call ID and forking related statistics for an active forked call.

Anchor Leg Call-id | Displays an internal anchor leg ID, which is the dial peer where forking enabled. The output displays the participant number and stream type. Stream type voice-near end indicates the called party side.

Non-Anchor Call-id | Displays an internal non-anchor leg ID, which is the dial peer where forking is not enabled. The output displays the participant number and stream type. Stream type voice-near end indicates the called party side.

Forked Call-id | This forking leg call-id will show near-end and far-end stream call-id details with state of the Stream.
Displays an internal foked leg ID. The output displays near-end and far-end details of a stream.

### Step 5

**show voip rtp forking**

Displays RTP media-forking connections.

**Example:**

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

### Output Field | Description
---|---
remote ip 10.42.29.7, remote port 38526, local port 18648 | Recording server IP, recording server port, and local CUBE device port where data for stream 1 was first sent from.

remote ip 10.42.29.7, remote port 50482, local port 17780 | Recording server IP, recording server port, and local CUBE device port where data for stream 2 was first sent from.

packets sent 29686 | Number of packets sent to the recorder.
### Output Field

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g711ulaw</td>
<td>Codec negotiated for the recording leg.</td>
</tr>
</tbody>
</table>

### Step 6
**show call active voice compact**

Displays a compact version of voice calls in progress. An additional call leg is displayed for media forking.

**Example:**

```
Device# show call active voice compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 3
  140 ANS T644 g711ulaw VOIP P10000 10.42.30.32:18638
  141 ORG T644 g711ulaw VOIP P708090 10.42.30.189:26184
  145 ORG T643 g711ulaw VOIP P595959 10.42.29.7:38526
```

### Step 7
**show call active video compact**

Displays a compact version of video calls in progress.

**Example:**

```
Device# show call active video compact
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 3
  1 ANS T14 H264 VOIP-VIDEO P1777 10.104.8.94:20036
  2 ORG T14 H264 VOIP-VIDEO P1888 10.104.8.98:29652
  6 ORG T13 H264 VOIP-VIDEO P1234 10.104.105.232:39318
```

### Step 8
**show sip-ua calls**

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

**Example:**

```
Device# show sip-ua calls
Total SIP call legs:3, User Agent Client:2, User Agent Server:1
SIP UAC CALL INFO
Call 1
SIP Call ID : 99EA5118-506211E0-80C6E01B-4C27AA62@10.42.30.10
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 10000
Called Number : 708090
Bit Flags : 0xC04018 0x10000100 0x80
CC Call ID : 141
Source IP Address (Sig ) : 10.42.30.10
Destn SIP Req Addr:Port : [10.42.30.5]:5060
Destn SIP Resp Addr:Port: [10.42.30.5]:5060
Destination Name : 10.42.30.5
Number of Media Streams : 1
Number of Active Streams: 1
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 141
Stream Type : voice+dtmf (1)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : rtp-nte
```
Dtmf-relay Payload Type : 101
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [10.42.30.10]:19256
Media Dest IP Addr:Port : [10.42.30.189]:26184
Options-Ping ENABLED: NO ACTIVE: NO
Call 2
SIP Call ID : 9A6D8922-506211E0-80CEE01B-4C27AA62@10.42.30.10
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Called Number : 595959
Bit Flags : 0xC04018 0x10800100 0x80
CC Call ID : 145
Source IP Address (Sig): 10.42.30.10
Destn SIP Req Addr:Port : [10.42.29.7]:5060
Destn SIP Resp Addr:Port: [10.42.29.7]:5060
Destination Name : 10.42.29.7
Number of Media Streams : 2
Number of Active Streams: 2
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 145
Stream Type : voice-nearend (3)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [10.42.30.10]:18648
Media Dest IP Addr:Port : [10.42.29.7]:38526
Media Stream 2
State of the stream : STREAM_ACTIVE
Stream Call ID : 146
Stream Type : voice-farend (5)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [10.42.30.10]:17780
Media Dest IP Addr:Port : [10.42.29.7]:50482
Options-Ping ENABLED: NO ACTIVE: NO
Number of SIP User Agent Client (UAC) calls: 2
SIP UAS CALL INFO
Call 1
SIP Call ID : 7CF44DF3-506611E0-8ED2B9D4-CA68C314@10.42.30.32
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Called Number : 10000
Bit Flags : 0x8C4401C 0x10000100 0x4
CC Call ID : 140
Source IP Address (Sig ): 10.42.30.10
Destn SIP Req Addr:Port : [10.42.30.32]:5060
Destn SIP Resp Addr:Port: [10.42.30.32]:52757
Destination Name : 10.42.30.32
Number of Media Streams : 1
Number of Active Streams: 1
RTP Fork Object : 0x0
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 140
Stream Type : voice+dtmf (0)
Stream Media Addr Type : 1
Negotiated Codec : g711ulaw (160 bytes)
Codec Payload Type : 0
Negotiated Dtmf-relay : rtp-nte
Dtmf-relay Payload Type : 101
QoS ID : -1
Local QoS Strength : BestEffort
Negotiated QoS Strength : BestEffort
Negotiated QoS Direction : None
Local QoS Status : None
Media Source IP Addr:Port: [10.42.30.10]:18792
Media Dest IP Addr:Port : [10.42.30.32]:18638
Options-Ping ENABLED:NO ACTIVE:NO
Number of SIP User Agent Server(UAS) calls: 1

Step 9
show call active video brief
Displays a truncated version of video calls in progress.

Example:
Device# show call active video brief
Telephony call-legs: 0
SIP call-legs: 3
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Total call-legs: 3

Step 10
debug ccsip messages (for audio calls)
Sent:
INVITE sip:22222@10.42.30.7:5060 SIP/2.0
Via: SIP/2.0/TCP 10.42.30.10:5060;branch=z9hG4bKv622CF
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:10.42.30.10>
Date: Fri, 18 Mar 2011 07:01:50 GMT
Call-ID: 6E6CF813-506411E0-80EAE01B-4C27A63316
Supported: 100rel, timer, resource-priority, replaces, sdp-anat
Min-SE: 1800
Cisco-Guid: 1334370502-1348997600-2396699092-3395863316
Verifying the Network-Based Recording Using CUBE

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;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;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
### Network-Based Recording

#### Verifying the Network-Based Recording Using CUBE

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

### Step 11

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

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

```
 Via: SIP/2.0/UDP 9.41.36.41:5060;branch=z9hG4bK2CC2408
 X-Cisco-Recording-Participant: sip:1777@10.104.45.207;media-index="0 2"
 X-Cisco-Recording-Participant: sip:1888@10.104.45.207;media-index="1 3"
 Cisco-Guid: 0884935168-0000065536-0000000401-3475859466
 v=0
 m=audio 17232 RTP/AVP 0 19
 a=sendonly
 m=audio 17234 RTP/AVP 0 19
 a=sendonly
 m=video 17236 RTP/AVP 126
```
Verifying the Network-Based Recording Using CUBE

<table>
<thead>
<tr>
<th>Output Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent: INVITE sip:575757@9.45.38.39:7686 SIP/2.0</td>
<td>22222 is the destination pattern or the number of recording server and is configured under the recorder dial peer.</td>
</tr>
<tr>
<td>X-Cisco-Recording-Participant: sip:1777@10.104.45.207;media-index=&quot;0 2&quot;</td>
<td>Cisco proprietary header with originating and terminating participant number and IP address used to communicate to the recording server.</td>
</tr>
<tr>
<td>X-Cisco-Recording-Participant: sip:1888@10.104.45.207;media-index=&quot;1 3&quot;</td>
<td></td>
</tr>
<tr>
<td>Cisco-Guid: 0884935168-0000065536-0000000401-3475859466</td>
<td>GUID is the same for the primary call and forked call.</td>
</tr>
<tr>
<td>m=audio 17232 RTP/AVP 0 19</td>
<td>First m-line of participant with payload type and audio codec.</td>
</tr>
<tr>
<td>m=audio 17234 RTP/AVP 0 19</td>
<td>Second m-line of another participant with payload type and audio codec.</td>
</tr>
<tr>
<td>m=video 17236 RTP/AVP 126</td>
<td>Third m-line of participant with video payload type and codec info.</td>
</tr>
<tr>
<td>m=video 17238 RTP/AVP 126</td>
<td>Fourth m-line of another participant with video payload type and codec info.</td>
</tr>
<tr>
<td>a=sendonly</td>
<td>CUBE is always in send only mode towards Recording server.</td>
</tr>
</tbody>
</table>

Receive:
SIP/2.0 200 OK

v=0
m=audio 1592 RTP/AVP 0
a=recvonly
m=audio 1594 RTP/AVP 0
a=recvonly
m=video 1596 RTP/AVP 126
a=recvonly

m=video 1598 RTP/AVP 126
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-FFFFFFFFB6661C0609.41.36.41
Max-Forwards: 70
CSeq: 101 ACK
Allow-Events: telephone-event
Content-Length: 0

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

**Step 12**

**debug ccsip messages** (for audio-only recording in a call with both audio and video)
Displays offer sent to MediaSense having only audio m-lines, when the **media-type audio** command is configured.

Sent:
INVITE sip:54321@9.45.38.39:36212 SIP/2.0
Via: SIP/2.0/UDP 9.41.36.15:5060;branch=z9hG4bK2216B
X-Cisco-Recording-Participant: sip:4321@9.45.38.39;media-index="0"
X-Cisco-Recording-Participant: sip:1111000010@9.45.38.39;media-index="1"
From: <sip:9.41.36.15>;tag=A2C74-5D9
To: <sip:54321@9.45.38.39>......
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length: 337

v=0
o=CiscoSystemsSIP-GW-UserAgent 9849 5909 IN IP4 9.41.36.15
s=SIP Call
c=IN IP4 9.41.36.15
t=0 0
Verifying the Network-Based Recording Using CUBE

Response from CUBE has inactive video m-lines.

Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.41.36.15:5060;branch=z9hG4bK2216B
.....
v=0

Step 13 Enter one of the following:

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

Displays detailed debug messages.

For Audio:
Media forking initialized:


Media forking started:

• 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/sipSPIUsValidCcb:
• Jun 15 10:37:55.404: //103/3E7E90AE8006/SIP/Function/ccsip_is_valid_ccb:
Verifying the Network-Based Recording Using CUBE

Network-Based Recording

Verifying the Network-Based Recording Using CUBE


Verifying the Network-Based Recording Using CUBE

Verifying the Network-Based Recording Using CUBE

Verifying the Network-Based Recording Using CUBE

Verifying the Network-Based Recording Using CUBE

Verifying the Network-Based Recording Using CUBE

Verifying the Network-Based Recording Using CUBE
Verifying the Network-Based Recording Using CUBE


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 16:40:01.787 IST* /522/34BF0A000000/SIP/Function/sipSPIGetMainStream:

*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/info/33792/ccsip_get_recording_participant_header: MF: X-Cisco header is PAI.

Adding a Video 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/32800/ccsip_ipip_media_forking_BuildAudioRecStream: MF: dtmf is inband

Video forking:

*Mar 19 16:40:01.789 IST* /522/34BF0A000000/SIP/Function/sipSPIGetVideoStream:

For Video
### Additional References for Network-Based Recording

#### Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediaSense Installation and Administration Guide</td>
<td>Cisco MediaSense Installation and Administration Guide</td>
</tr>
</tbody>
</table>

#### Standards and RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 3984</td>
<td>RTP Payload Format for H.264 Video</td>
</tr>
<tr>
<td>RFC 5104</td>
<td>Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)</td>
</tr>
<tr>
<td>RFC 5168</td>
<td>XML Schema for Media Control</td>
</tr>
</tbody>
</table>