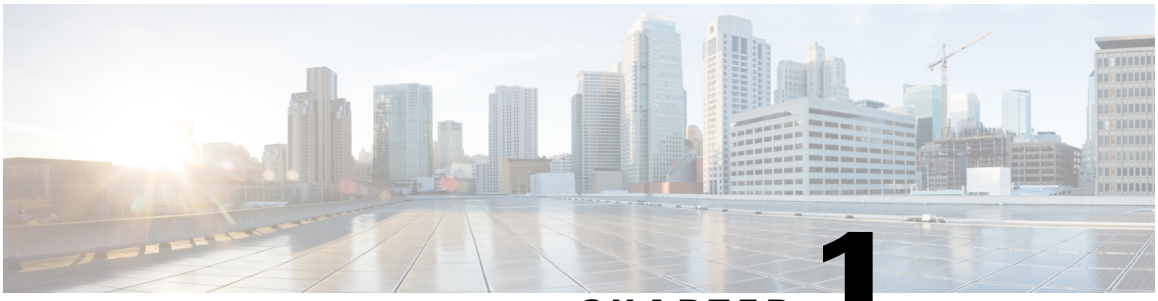
Cisco MediaSense Solution Reference Network Design Guide, Release 10.0(1)

First Published: December 12, 2013

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

CONTENTS



CHAPTER 1

Product overview

Cisco MediaSense is a SIP-based, network-level service that provides voice and video media recording capabilities for other network devices. Fully integrated into Cisco's Unified Communications architecture, MediaSense automatically captures and stores every Voice over IP (VoIP) conversation which traverses appropriately configured Unified Communications Manager IP phones or Cisco Unified Border Element (CUBE) devices. In addition, an IP phone user or SIP endpoint device may call the MediaSense system directly in order to leave a recording consisting of media generated only by that user. Such recordings can include video as well as audio—offering a simple and easy method for recording video blogs and podcasts.

Since forked media can be recorded from either a Cisco IP phone or a CUBE device, MediaSense allows you to record a conversation from different perspectives. Recordings forked by an IP phone are treated from the perspective of the phone itself—any media flowing to or from that phone gets recorded. If the call gets transferred to another phone however, the remainder of the conversation does not get recorded (unless the target phone has recording enabled as well). This perspective may work well for contact center supervisors whose focus is on a particular agent.

Recordings forked by CUBE are treated from the perspective of the caller. All media flowing to or from the caller gets recorded, no matter how many times the call gets transferred inside the enterprise. Even interactions between the caller and an Interactive Voice Response (IVR) system where no actual phone is involved will be recorded. The only part of the call which will not be recorded would be a consult call from one IP phone to another— for example, as part of a consult transfer. (Even that can be recorded if Unified Communications Manager is configured to route IP phone to IP phone calls through a CUBE.) This perspective may work well for dispute resolution or regulatory compliance purposes, where the focus is on the caller.

No matter how they are captured, recordings may be accessed in several ways. While a recording is still in progress, it can be streamed live ("monitored") through a computer which is equipped with a media player such as VLC or RealPlayer, or one provided by a partner or 3rd party. Once completed, recordings may be played back in the same way, or downloaded in raw form via HTTP. They may also be converted into .mp4 or .wav files and downloaded in that format. All access to recordings, either in progress or completed, is through web-friendly URIs. MediaSense also offers a web-based Search and Play application with a built-in media player. This allows authorized users to select individual calls to monitor, playback, or download directly from a supported web browser.

In addition to its primary media recording functionality, MediaSense offers two other capabilities.

It can play back specific video media files on demand on video phones or supported players. This capability supports Video in Queue (ViQ), Video on Demand (VoD), or Video on Hold (VoH) use cases in which a separate call controller invites MediaSense into an existing video call in order to play a previously designated recording. An administrator can upload studio-recorded videos in MP4 format and then configure individual

incoming dialed numbers to automatically play those uploaded videos. The call controller plays the video by sending a SIP invitation to MediaSense at the dialed number.

MediaSense can also integrate with Cisco Unity Connection to provide video voice-mail greetings. Videos are recorded on MediaSense directly by Unity Connection subscribers and are then played back to their video-capable callers before they leave their messages.

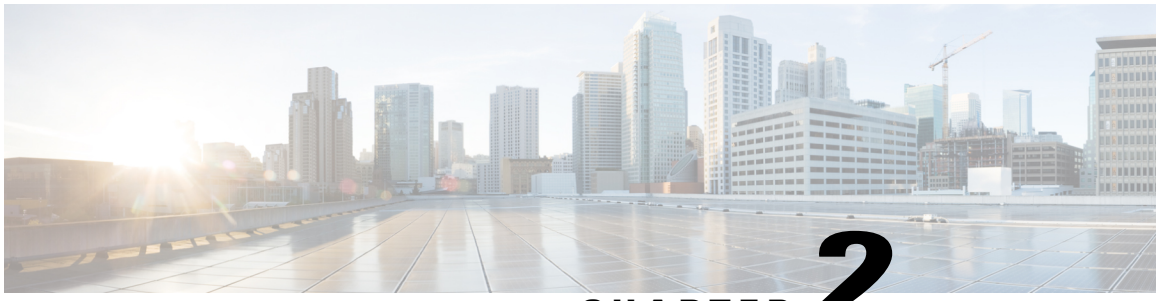
Media recordings occupy a fair amount of disk space, so space management is a significant concern. MediaSense offers two modes of operation with respect to space management: retention priority and recording priority. These modes address two opposing and incompatible use cases; one where all recording sessions must be retained until explicitly deleted (even if it means new recording sessions cannot be captured) and one where older recording sessions can be deleted if necessary to make room for new ones. A sophisticated set of events and APIs is provided for client software to automatically control and manage disk space.

MediaSense also maintains a metadata database where information about all recordings is maintained. A comprehensive Web 2.0 API is provided that allows client equipment to query and search the metadata in various ways, to control recordings that are in progress, to stream or download recordings, to bulk-delete recordings that meet certain criteria, and to apply custom tags to individual recording sessions. A Symmetric Web Services (SWS) eventing capability enables server-based clients to be notified when recordings start and stop, when disk space usage exceeds thresholds, and when meta-information about individual recording sessions is updated. Clients may use these events to keep track of system activities and to trigger their own actions.

Taken together, these MediaSense capabilities target four basic use cases:

1. Recording of conversations for regulatory compliance purposes (compliance recording).
2. Capturing or forwarding media for transcription and speech analytics purposes.
3. Capturing of individual recordings for podcasting and blogging purposes (video blogging).
4. Playing back previously uploaded videos for ViQ, VoD, VoH, or video voice-mail greeting purposes.

Compliance recording may be required in any enterprise, but is of particular value in contact centers where all conversations conducted on designated agent phones or all calls from customers must be captured and retained and where supervisors need an easy way to find, monitor, and play conversations for auditing, training, or dispute resolution purposes. Speech analytics engines are well served by the fact that MediaSense maintains the two sides of a conversation as separate tracks and provides access to each track individually, greatly simplifying the analytics engine need to identify who is saying what.



CHAPTER 2

Characteristics and features

This section provides design-level details on compliance recording, direct inbound and outbound recording, and monitoring using MediaSense.

Compliance recording

In compliance recording, calls are configured to always be recorded.

For IP phone recording, all calls received by or initiated by designated phones are recorded. Individual lines on individual phones are enabled for recording by configuring them with an appropriate recording profile in Unified Communications Manager.

For CUBE recording, all calls passing through the CUBE that match particular dial peers (typically selected by dialed number pattern) are recorded. MediaSense itself does not control which calls are recorded (except to the limited extent described under Incoming call handling rules).

Compliance recording differs from selective recording because in selective recording, the recording server determines which calls it will record. MediaSense itself does not support selective recording, but the effect can be achieved by deploying MediaSense in combination with certain partner applications.

Recording is accomplished by *media forking*, where basically the phone or CUBE sends a copy of the incoming and outgoing media streams to the MediaSense recording server. When a call originates or terminates at a recording-enabled phone, Unified Communications Manager sends a pair of SIP invitations to both the phone and the recording server. The recording server prepares to receive a pair of real-time transport protocol (RTP) streams from the phone. Similarly, when a call passes through a recording-enabled CUBE, the CUBE device sends a SIP invitation to the recording server and the recording server prepares to receive a pair of RTP streams from the CUBE.

This procedure has several implications:

- Each recording session consists of two media streams (one for media flowing in each direction). These two streams are captured separately on the recorder, though both streams (or *tracks*) end up on the same MediaSense recording server.
- Most, but not all, Cisco IP phones support media forking. Those which do not support media forking cannot be used for phone-based recording.
- Though the phones can fork copies of media, they cannot transcode. This means that whatever codec is negotiated by the phone during its initial call setup, is the codec used in recording. MediaSense supports a limited set of codecs; if the phone negotiates a codec which is not supported by MediaSense, the call will not be recorded. The same is true for CUBE recordings.
- The recording streams are set up only after the phone's primary conversation is fully established, which could take some time to complete. Therefore, there is a possibility of clipping at the beginning of each call. Clipping is typically limited to less than two seconds, but it can be affected by overall CUBE, Unified Communications Manager, and MediaSense load; as well as by network performance characteristics along the signaling link between CUBE or Unified Communications Manager and MediaSense. MediaSense carefully monitors this latency and raises alarms if it exceeds certain thresholds.

MediaSense does not initiate compliance recording. It only receives SIP invitations from Unified Communications Manager or CUBE and is not involved in deciding which calls do or do not get recorded. The IP phone configuration and the CUBE dial peer configuration determine whether media should be recorded. In some cases, calls may be recorded more than once, with neither CUBE, Unified Communications Manager, nor MediaSense being aware that it is happening.

This would be the case if, for example, all contact center agent IP phones are configured for recording and one agent calls another agent. It might also happen if a call passes through a CUBE which is configured for recording and lands at a phone which is also configured for recording. The CUBE could end up creating two recordings of its own. However, MediaSense stores enough metadata that a client can invoke a query to locate duplicate calls and selectively delete the extra copy.

At this time, only audio streams can be forked by Cisco IP phones and CUBE. Compliance recording of video media is not supported; it is only available for the blogging modes of recording. CUBE is capable of forking the audio streams of a video call and MediaSense can record those, but video-enabled Cisco IP phones do not offer this capability.

MediaSense can record calls of up to eight hours in duration.

Conferences and transfers

MediaSense recordings are made up of one or more sessions where each media forking session contains two media streams—one for incoming and one for the outgoing data. A simple call consisting of a straightforward two-party conversation is represented entirely by a single session. MediaSense uses metadata to track which participants are recorded in which track of the session, as well as when they entered and exited the conversation—but it cannot always do so when conferences are involved.

When sessions included transfer and conference activities, MediaSense does its best to retain the related information in its metadata. If a recording gets divided into multiple sessions, metadata is also available to help client applications correlate those sessions together.

Conferences

A multi-party conference is also represented by a single session with one stream in each direction, with the conference bridge combining all but one of the parties into a single MediaSense participant. There is metadata to identify that one of the streams represents a conference bridge, but MediaSense does not receive the full list of parties on the conference bridge.

Transfers

Transfers behave differently depending on whether the call is forked from a Unified Communications Manager phone or from a CUBE.

With Unified Communications Manager recordings, the forking phone anchors the recording. Transfers that drop the forking phone terminate the recording session but transfers that keep the forking phone in the conversation do not.

With CUBE forking, the situation is more symmetric. CUBE is an intermediary network element and neither party is an anchor. Transfers on either side of the device are usually accommodated within the same recording session. (See Solution-level deployment models for more information.)

Hold and pause

Hold and pause are two concepts sound similar, but they are not the same.

- Hold (and resume) takes place as a result of a user pressing a key on his or her phone. MediaSense is a passive observer.
- Pause (and resume) takes place as a result of a client application issuing a MediaSense API request to temporarily stop recording while the conversation continues.

Hold behavior differs depending on which device is forking media. In Unified Communications Manager deployments, one party places the call on hold, blocking all media to or from that party's phone while the other phone typically receives music (MOH). If the forking phone is the one that invokes the hold operation, Unified Communications Manager terminates the recording session and creates a new recording session once the call is resumed. Metadata fields allow client applications to gather together all of the sessions in a given conversation.

If the forking phone is not the one that invokes the hold operation, the recording session continues without a break and even includes the music on hold—if it is unicast (multicast MOH does not get recorded).

For deployments where Unified Communications Manager phones are configured for selective recording, there must be a CTI (TAPI or JTAPI) client that proactively requests Unified Communications Manager to begin recording any given call. The CTI client does not need to retrigger recording in the case of a hold and resume.

For CUBE deployments, hold and resume are implemented as direct SIP operations and the SIP protocol has no direct concept of hold and resume. Instead, these operations are implemented in terms of media stream inactivity events. MediaSense captures these events in its metadata and makes it available to application clients, but the recording session continues uninterrupted.

The **Pause** feature allows applications such as Customer Relationship Management (CRM) systems or VoiceXML-driven IVR systems to automatically suppress recording of sensitive information based on the caller's position in a menu or scripted interaction. Pause is invoked by a MediaSense API client to temporarily stop recording, and the subsequent playback simply skips over the paused segment. MediaSense does store the information in its metadata and makes it available to application clients.

Pause behaves identically for CUBE and Unified Communications Manager recording.

Direct inbound recording

In addition to compliance recording controlled by a CUBE or a Unified Communications Manager recording profile, recordings can be initiated by directly dialing a number associated with a MediaSense server configured for automatic recording. These recordings are not carried out through media forking technology and therefore are not limited to CUBE or Cisco IP phones, nor are they limited to audio media. This is how video blogging is accomplished.

Direct outbound recording

Using the MediaSense API, a client requests MediaSense to call a phone number. When the recipient answers, the call is recorded similarly to the way it is recorded when a user dials the recording server in a direct Inbound call. The client can be any device capable of issuing an HTTP request to MediaSense, such as a 'call me' button on a web page. Any phone, even a non-IP phone (like a home phone), can be recorded if it is converted to IP using a supported codec. Supported IP video phones can also be recorded in this way.

Direct outbound recording is only supported if MediaSense can reach the target phone number through a Unified Communications Manager system. In CUBE-only deployments where Unified Communications Manager is not used for call handling, direct outbound recording is not supported.

Monitoring

While a recording is in progress, the session is monitored by a third-party streaming-media player or by the built-in media player in MediaSense.

To monitor a call from a third-party streaming-media player, a client must specify a real time streaming protocol (RTSP) URI that is prepared to supply HTTP-BASIC credentials and is capable of handling a 302 redirect. The client can obtain the URI either by querying the metadata or by capturing session events.

MediaSense offers an HTTP query API that allows suitably authenticated clients to search for recorded sessions based on many criteria, including whether the recording is active. Alternatively, a client may subscribe for session events and receive MediaSense Symmetric Web Service (SWS) events whenever a recording is started

(among other conditions). In either case, the body passed to the client includes a great deal of metadata about the recording, including the RTSP URI to be used for streaming.

The third-party streaming-media players that Cisco has tested for MediaSense are VLC and RealPlayer. Each of these players has advantages and disadvantages that should be taken into account when selecting which one to use.

Recording sessions are usually made up of two audio tracks. MediaSense receives and stores them that way and does not currently support real time mixing.

VLC is capable of playing only one track at a time. The user can alternate between tracks but cannot hear both simultaneously. VLC is open source and is easy to embed into a browser page.

RealPlayer can play the two streams as stereo (one stream in each ear) but its buffering algorithms for slow connections sometimes results in misleading periods of silence for the listener. People are more or less used to such delays when playing recorded music or podcasts, but call monitoring is expected to be real time and significant buffering delays are inappropriate for that purpose.

None of these players can render AAC-LD, g.729 or g.722 audio. A custom application must be created in order to monitor or play streams in those forms.

MediaSense's built-in media player is accessed by a built-in Search and Play application. This player covers more codecs and can play both streams simultaneously, but it cannot play video, and it cannot support the AAC-LD codec. This applies to both playback of recorded calls and monitoring of active calls.

Only calls that are being recorded are available to be monitored. Customers who require live monitoring of unrecorded calls, or who cannot accept these other restrictions, may wish to consider Unified Communications Manager's Silent Monitoring capability instead.

Playback

Once a recording session has completed, it can be played back on a third-party streaming-media player or through the built-in media player in the Search and Play application. Playing it back through a third-party streaming-media player is similar to monitoring—an RTSP URI must first be obtained either through a query or an event.

Silence suppression

While recording a call, it is possible to create one or more segments of silence within the recording (for example by invoking the pauseRecording API). Upon playback, there are various ways to represent that silence. The requesting client uses a set of custom header parameters on the RTSP PLAY command to specify one of the following:

1. The RTP stream pauses for the full silent period, then continues with a subsequent packet whose mark bit is set and whose timestamp reflects the elapsed silent period.
2. The RTP stream does not pause. The timestamp reflects the fact that there was no pause, but the RTP packets contain "TIME" padding which includes the absolute UTC time at which the packet was recorded.
3. The RTP stream compresses the silent period to roughly half a second; in all other respects it acts exactly like bullet 1. This is the default behavior and is how the built-in media player works.

In all cases, the file duration returned by the RTSP DESCRIBE command reflects the original record time duration. It is simply the time the last packet ended minus the time the first packet began.

The session duration returned by the MediaSense API and session events may differ because these are based on SIP activity rather than on media streaming activity.

Commercial media players such as VLC and RealPlayer elicit the default behavior described in bullet 3. However, these players are designed to play music and podcasts, they are not designed to handle media streams that include silence—so they may hang, disconnect, or not seek backwards and forwards in the stream.

Conversion and download

Completed recording sessions can be converted on demand to .mp4 or .wav format via an HTTP request. Files converted this way format carry two audio tracks—not as a mixed stream, but as stereo. Alternatively, .mp4 files can also carry one audio and one video track.

After conversion, .mp4 and .wav files are stored for a period of time in MediaSense along with their raw counterparts and are accessible using their own URLs. (The files eventually get cleaned up automatically, but are recreated on demand the next time they are requested.) As with streaming, browser or server-based clients can get the URIs to these files by either querying the metadata or monitoring recording events. The URI is invoked by the client to play or download the file.

As with RTSP streaming, the client must provide HTTP-BASIC credentials and be prepared to handle a 302 redirect. In this way, conversion to .mp4 or .wav format provides a secure, convenient, and standards-compliant way to package and export recorded sessions.

However, large scale conversion to .mp4 or .wav takes a lot of processing power on the recording server and may impact performance and scalability. To meet the archiving needs of some organizations, as well as to serve the purposes of those speech analytics vendors who would rather download recordings than stream them in real time, MediaSense offers a "low overhead" download capability.

This capability allows clients using specific URIs to download unmixed and unpackaged individual tracks in their raw g.722, g.711 or g.729 format. The transport is HTTP 1.1 chunked, which leaves it up to the client (and the developer's programming expertise) to reconstitute and package the media into whatever format best meets its requirements. As with the other retrieval methods, the client must provide HTTP-BASIC credentials and be prepared to handle a 302 redirect. Note that video streams and AAC-LD encoded audio streams cannot currently be downloaded in this way.

Embedded Search and Play application

MediaSense provides a web-based tool used to search, download, and playback recordings. This Search and Play application is accessed using the API user credentials.

The tool searches both active and past recordings based on metadata characteristics such as time frame and participant extension. Recordings can also be selected using call identifiers such as Cisco-GUID or Unified CM call leg identifier. Once recordings are selected, they may be individually downloaded in mp4 or .wav format or played using the application's built-in media player.

The Search and Play tool is built using the MediaSense REST-based API. Customers and partners interested in building similar custom applications can access this API from the DevNet (formerly known as the Cisco Developer Network).

Support for the Search and Play application is limited to clusters with a maximum of 400,000 sessions in the database. Automatic pruning provides the capability to adjust the retention period to ensure that this limitation is respected using the following formula:

Retention Setting in Days = 400,000 / (avg # agents * avg # calls per hour * avg # hours per day)

For example, if you have 100 agents taking 4 calls per hour, 8 hours per day every day, you can retain these sessions for 125 days before exceeding the 400,000 session limit. This is acceptable for most customers, but

if you have 1000 agents taking 30 calls per hour, 24 hours per day every day, your retention period is about half a day. The Search and Play application cannot be used in this kind of environment.

Note Additional reasons for limiting the retention period are described in Scalability and sizing.

Embedded streaming media player

Telephone recording uses a different set of codecs than those typically used for music and podcasts. As a result, most off-the-shelf media players are not well suited to playing the kind of media that MediaSense records. This is why partner applications generally provide their own media players, and why MediaSense has the built-in Search and Play application.

The embedded player supports g.729, g.711, and g.722 codecs. This applies to both playback of recorded calls and monitoring of active calls.

The embedded media player can be accessed through the Search and Play application or it can be used by a 3rd party client application. Such an application can present a clickable link to the user that, when clicked, loads the recording-specific media player for the selected recording session into the user's browser. This allows partners who do not have sophisticated user interface requirements to avoid the complexity of either developing their own media player or incorporating an off the shelf media player into their applications.

Uploaded videos to support ViQ, VoD and VoH features

MediaSense supports the Cisco Contact Center Video in Queue, Video on Demand, and Video on Hold features by enabling administrators to upload .mp4 video files for playback on demand.

To use these features, users must:

1. Produce an .mp4 video that meets the technical specifications outlined below.
2. Upload the .mp4 video to the MediaSense Primary node. The video is automatically converted into a form that can be played back to a supported video endpoint and distributed to all other nodes. Playback is automatically load balanced across the cluster.
3. Create an "incoming call handling rule" that maps a particular incoming dialed number to the uploaded video. You may also specify whether this video should be played once or repeated continuously.

Administrative user interfaces are provided for uploading the file to MediaSense and creating the incoming call handling rule. These functions are not available through the MediaSense API.

An .mp4 file is a container that may contain many different content formats. MediaSense requires that the file content meet the following specifications:

- The file must contain one audio track and one video track.
- The video must be encoded using the H.264.
- The audio must be encoded using AAC-LC.
- The audio must be monaural.
- The entire .mp4 file size must not exceed 2GB.

The preceding information is known as the *Studio Specification*. It must be provided to any professional studio that is producing video content for this purpose. Most commonly available consumer video software products can also produce this format.

Note Video resolution and aspect ratio are not enforced by MediaSense. MediaSense will play back whatever resolution it finds in an uploaded file, so it is important to use a resolution that looks good on all the endpoints on which you expect the video to be played. Many endpoints are capable of up- or down-scaling videos as needed, but some (such as the Cisco 9971) are not. For the best compatibility with all supported endpoints, use standard VGA resolution (640x480).

Cisco endpoints do not support AAC-LC audio (which is the standard for .mp4), so MediaSense automatically converts the audio to AAC-LD, g.711 µlaw, and g.722 (note that g.711aLaw is not supported for ViQ/VoH). MediaSense automatically negotiates with the endpoint to determine which audio codec is most suitable. If MediaSense is asked to play an uploaded video to an endpoint which supports only audio, then only the audio track will be played.

Video playback capability is supported on all supported MediaSense platforms, but there are varying capacity limits on some configurations. See the "Hardware profiles" section below for details.

MediaSense comes with a sample video pre-loaded and pre-configured for use directly out of the box. After successful installation or upgrade, dial the SIP URL **sip:SampleVideo@<mediasense-hostname>** from any supported endpoint or from Cisco Jabber Video to see the sample video.

Integration with Unity Connection for video voice-mail

Beginning with Cisco Unity Connection (CUC) release 10.0(1), configured subscribers have the option to record video greetings in addition to audio greetings. Subscribers who are configured to record video greetings and who are calling from a video capable IP endpoint are presented with additional prompts to record their video greeting. These recordings (both the audio and video tracks) are stored and played back from MediaSense. A separate audio-only copy of the recording remains on Unity Connection as well.

If for any reason Unity Connection is not able to play a video greeting from MediaSense, it reverts to its locally stored audio greeting.

This is an introductory implementation and therefore contains a number of limitations.

- A single, dedicated MediaSense node may be connected to one Unity Connection node, where a node is a single instance of CUC or an HA pair.

  The MediaSense node may not be used for any other MediaSense purpose.
- The scale is limited to approximately 35 simultaneous video connections.
- Cisco 9971 and similar phones are supported and they must be configured to support g.711 audio. See http://www.cisco.com/en/US/docs/voice_ip_comm/connection/10x/design/guide/10xcucdg070.html for a detailed list of supported phones.

More information about the Cisco Unity Connection integration, including deployment and configuration instructions, can be found in the Unity Connection documentation.

Integration with Finesse and Unified CCX

MediaSense is integrated with Cisco Finesse and Unified Contact Center Express (Unified CCX). The integration is both at the desktop level and at the MediaSense API level.

At the desktop level, MediaSense's Search and Play application has been adapted to work as an OpenSocial gadget that can be placed on a Finesse supervisor's desktop. In this configuration, MediaSense can be configured to authenticate against Finesse rather than against Unified CM. Therefore, any Finesse user who has been assigned a supervisor role can search and play recordings from MediaSense directly from his or her Finesse desktop. (A special automatic sign-on has been implemented so that when the supervisor signs in to Finesse, he or she is also automatically signed into the MediaSense Search and Play application.) Note that other than this sign-in requirement, there are currently no constraints on access to recordings. Any Finesse supervisor has access to any and all recordings.

At the API level, Unified CCX subscribes for MediaSense recording events and matches the participant information it receives with the agent extensions that it knows about. It then immediately tags those recordings in MediaSense with the agentId, teamId, and if it was an ICD call, the contact service queue identifier (CSQId) of the call. This allows the supervisor, through the Search and Play application, to find recordings which are associated with particular agents, teams, or CSQs without having to know the agent extensions.

This integration uses BiB forking, selectively invoked through JTAPI by Unified CCX. Because Unified CCX is in charge of starting recordings, it is also in charge of managing and enforcing Unified CCX agent recording licenses. However, other network recording sources (such as un-managed BiB forking phones or CUBE devices) could still be configured to direct their media streams to the same MediaSense cluster, which could negatively impact Unified CCX's license counting.

For example, Unified CCX might think it has 84 recording licenses to allocate to agent phones as it sees fit, but it may find that MediaSense is unable to accept 84 simultaneous recordings because other recording sources are also using MediaSense resources. This also applies to playback and download activities—any activity that impacts MediaSense capacity. If you are planning to allow MediaSense to record other calls besides those that are managed by Unified CCX, then it is very important to size your MediaSense servers accordingly.

More information about this integration, including deployment and configuration instructions, can be found in the Unified CCX documentation.

Integration with Unified CM for Video on Hold and native queuing

Starting with Unified CM Release 10.0, customers can configure a Video on Hold source for video callers, similar to a Music on Hold source that is used for audio callers. The same facility is used to provide pre-recorded video to callers who are waiting for a member of a hunt group to answer. This is known as "CUCM native queuing."

MediaSense can be used as the video media server for both purposes. To use MediaSense in this way, administrators make use of the product's generic ability to assign incoming dialed numbers to various uploaded videos, which are then played back when an invitation arrives on those dialed numbers. Unified CM causes one of these videos to play by temporarily transferring the call to the corresponding dialed number on MediaSense.

See Uploaded videos to support ViQ, VoD and VoH features for more information.

For instructions on configuring these features in Unified CM, see the relevant Unified CM documentation.

Integration with Cisco Remote Expert

MediaSense integrates with the Cisco Remote Expert product in two areas:

- It can act as a video media server for ViQ, VoH, and Video IVR.
- It can record the audio portion of the video call.

MediaSense's video media server capabilities satisfy Remote Expert's needs for ViQ, VoH, and Video IVR. See Uploaded videos to support ViQ, VoD and VoH features for more information.

Calls that are to be recorded must be routed through a CUBE device that is configured to fork its media streams to MediaSense (because most of the endpoints used for Remote Expert are not able to fork media themselves). All the codecs listed in Codecs supported are supported, except for the video codec, H.264. If your version of IOS does fork video along with the audio streams, MediaSense will only capture the audio. Please consult the Compatibility matrix to ensure that your CUBE is running a supported version of IOS, to ensure that you incorporate several bug fixes in this area.

Remote Expert provides its own user interface portal for finding and managing recordings, and for playing them back. For AAC-LD audio calls (most common when using EX-series endpoints), there are no known RTSP-based AAC-LD streaming media players, so those calls can only be converted to .mp4 and downloaded for playback. Live monitoring of such calls is not possible.

For more information about this integration, including deployment and configuration instructions, see the Remote Expert documentation.

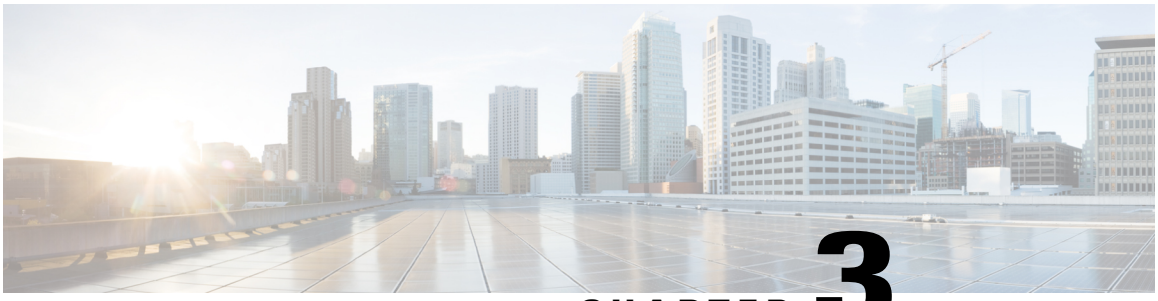
Incoming call handling rules

When MediaSense receives a call, it needs to know what action to take. In Releases 9.0(1) and earlier, all incoming calls would simply be recorded—irrespective of the dialed number to which the call was addressed. As of Release 9.1(1), you have the option to configure what action MediaSense takes for each call type. The following actions are available:

- Record the call.
- Reject the call.
- Play a specified uploaded video once.
- Play a specified uploaded video repetitively.

If your application is to record calls forked by a CUBE, then the dialed number in question is configured as the "destination-pattern" setting in the dial peer which points to MediaSense. If your application is to record calls forked by a Unified Communications Manager phone, then the dialed number in question is configured as the recording profile's route pattern.

For compatibility with earlier releases, all incoming addresses (except for SampleVideo) are configured to record.



CHAPTER 3

Codecs supported

Basic recordings

MediaSense can accept

- audio in
  - g.711 μLaw and aLaw
  - g.722
  - g.729, g.729a, g.729b
  - AAC-LD (also known as MP4A/LATM)
- and video in h.264 encoding.

Note that off-the-shelf streaming media players typically do not support the AAC-LD, g.722 and g.729 codecs, though the media player which is embedded in the built-in Search and Play application can support either g.722 or g.729 but neither it nor any commonly available media player can support AAC-LD. AAC-LD-based recordings must be converted to .mp4 or .wav format and played as downloaded files. Conversations that use AAC-LD cannot be monitored live.

Neither Unified Communications Manager nor CUBE performs a full codec negotiation with MediaSense. They negotiate codecs among the conversation endpoints first and then initiate a connection to MediaSense. If they happen to select a codec which is not supported by MediaSense, the call will not be recorded.

Therefore, for all phones that need to be recorded, it is important to configure them so that the codec that gets selected for the phones is the codecs that MediaSense supports.

For Unified Communications Manager recording, some of the newer Cisco IP phones support iLBC or iSAC. For those phones, Unified Communications Manager may prefer to negotiate them (if possible). However, since MediaSense does not accept these codecs, they must be disabled for recording enabled devices in Unified Communications Manager's service parameter settings.

MediaSense is capable of recording the audio portion of Telepresence calls among EX-90 and SX-20 devices when the conversation traverses a CUBE device. However, these endpoints must be configured to use a g.711(aLaw or μLaw), g.722, or AAC-LD codec.

Mid-call codec changes may be implemented based on call flow activities—most notably when a call is transferred or conferenced with a phone which has different codec requirements than those which were negotiated during the initial invitation. This is particularly common in CVP-based contact center deployments where a call may be queued at a VXML gateway playing g.711 music, and is then delivered to a g.729 agent.

The results of a mid-call codec change differ depending on whether CUBE or Unified Communications Manager is providing the forked media. With Unified Communications Manager forking, once the recording codec has been established, it cannot be changed. If a remote party transfers the call to a phone which cannot accept the previously selected codec, then Unified Communications Manager tries to insert a transcoder between the two phones so that the recording codec can remain constant. If no transcoder is available, Unified Communications Manager drops the transferred call and terminates the recording.

With CUBE-based forking, the codec is allowed to change. If that happens, MediaSense terminates the existing recording session and begins a new one using the new codec. The conversation then appears in MediaSense in the form of two successive but separate sessions, with different sessionIds, but sharing the same CCID call identifier.

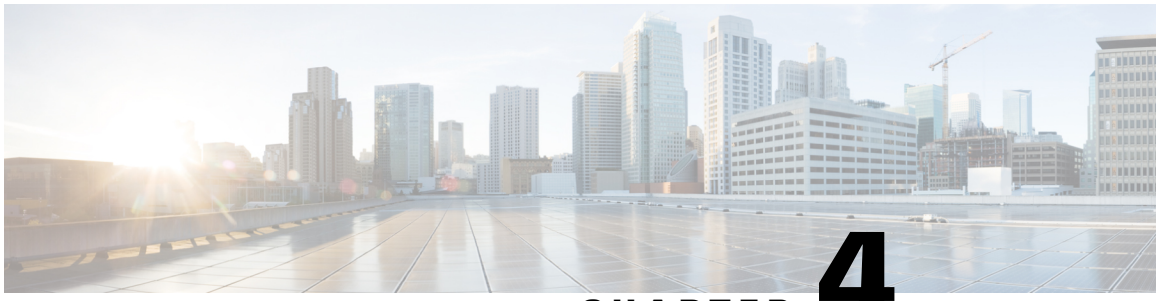
For both CUBE and Unified CM recording, it is not possible for the two audio tracks in a session to be assigned different codecs.

Video greetings

Video voice-mail greetings (used with Unity Connection integration) are designed to work only with Cisco 9971 (or similar) phones using g.711 uLaw or aLaw and with h.264 video. These greetings can only be played back on phones that support these codecs and the video resolution at which the greeting was recorded. When an incompatible phone reaches a video-enabled mailbox, the caller does not see the video portion of the greeting. See http://www.cisco.com/en/US/docs/voice_ip_comm/connection/10x/design/guide/10xcucdg070.html for a detailed list of supported phones.

Uploaded videos

Uploaded videos must be provided in .mp4 format using h.264 for video and AAC-LC for audio (see the exact Studio Specification below). The audio is converted to AAC-LD, g.711 µLaw (not aLaw), and g.722 for streaming playback. Most media players (including the built-in one) and most endpoints (including Cisco 9971 video phones, Jabber soft phones, and Cisco EX-60 and EX-90 Telepresence endpoints) can play at least one of these formats.



CHAPTER 4

Metadata database and the MediaSense API

MediaSense maintains a database containing a great deal of information about recorded sessions. The database is stored redundantly on the primary and secondary servers. The data includes:

- Various track, participant, call and session identifiers.
- Time stamps and durations.
- Real time session state.
- URIs for streaming and downloading recordings in various formats.
- Server address where recorded files are stored.

Tags

Along with the preceding information, MediaSense stores tags for each session.

Tags are brief, arbitrary, text strings that a client can specify and associate to individual sessions using the Web 2.0 APIs, and optionally, to specific time offsets within those sessions. Timed session tags are useful for identifying points in time when something happened, such as when a caller became irate or an agent gave erroneous information. Un-timed session tags may be used to attach notes which are applicable to the entire session, such as a contact center agent ID or to mark or categorize some sessions with respect to other sessions.

MediaSense also uses the tagging facility to mark when certain actions occurred during the session (such as pause and resume) or when the media inactivity state changes as reported by the SIP signaling. These are known as system-defined tags.

While most tags are associated with an entire session, media inactivity state change tags are associated with a specific track in the session.

MediaSense API

The MediaSense API offers a number of methods to search and retrieve information in the metadata database. Authenticated clients perform simple queries such as finding all sessions that have been deleted by the automatic pruning mechanism or finding all sessions within a particular time range that are tagged with a certain string. The API also supports much more complex queries as well as a sorting and paging scheme by which only a selected subset of the result set will be returned.

The API provides access to a number of other MediaSense functions as well. Use the API to subscribe for events, to manage disk storage, to manipulate recording sessions that are in progress, to remove unneeded inactive sessions and recover their resources, and to invoke operations such as conversion to .mp4 or .wav. Lengthy operations are supported through a remote batch job control facility. The API is described in detail in the Cisco MediaSense Developer Guide.

MediaSense API interactions are conducted entirely over HTTPS and require that clients be authenticated. Depending on the type of request, clients will use either POST or GET methods. Response bodies are always delivered in JSON format. HTTP version 1.1 is used, which allows TCP links to remain connected from request to request. For best performance, clients should be written to do the same.

API requests may be addressed to either the primary or the secondary server (the client needs to authenticate to each server separately), and must provide the HTTP session identifier that was previously obtained from the server being addressed.

Events

The MediaSense event mechanism provides server-based clients with immediate notification when actions of interest to them take place. The following types of events are supported:

- Session events - when recording sessions are started, ended, updated, deleted, or pruned.
- Tag events - when tags are attached to or removed from recorded sessions.
- Storage threshold events - when disk space occupancy rises above or falls below certain preconfigured thresholds.

Session events provide critical information about a session given its current state. A client could then, for example, use the URIs provided in these events to offer real time monitoring and control buttons to an auditor or contact center supervisor. A client might also implement a form of selective recording (as opposed to compliance recording) by deleting (after the fact) sessions that it determines do not need to be recorded.

Tag events are used as a form of inter-client communication: when a session is tagged by one client, all other subscribed clients are informed about it.

Storage threshold events allow a server-based client application to manage disk usage. The client would subscribe to these events and selectively delete older recordings (when necessary) according to its own rules. For example, the client might tag selected sessions for retention and then when a threshold event is received, delete all sessions older than a certain date except those tagged for retention.

Events are populated with JSON formatted payloads and delivered to clients using a Symmetric Web Services protocol (SWS), which is essentially a predefined set of HTTP requests sent from MediaSense to the client (note that HTTPS is not currently supported for eventing).

When a client subscribes for event notifications, it provides a URL to which MediaSense will address its events, as well as a list of event types or categories in which it has an interest. Any number of clients may

subscribe and clients may even subscribe on behalf of other recipients (i.e., the subscribing client may specify a host other than itself as the event recipient). The only restriction is that there cannot be more than one subscription to the same URL.

Events are always generated by either the primary or the secondary server. When both are deployed, each event is generated on one server or the other, but not both (which has implications for high availability).Customers must choose one of two modes of event delivery - one which favors reliability or one which favors convenience.

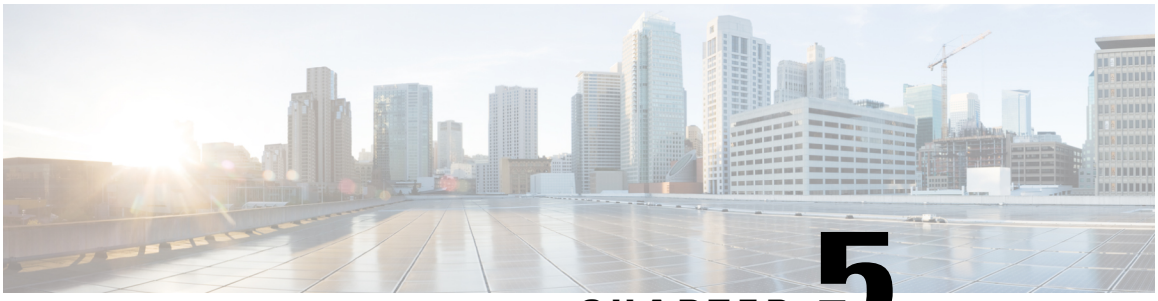
Metadata differences between CUBE and Unified Communications Manager

At a high level, the metadata which is captured by MediaSense is call controller agnostic. Every recording is made up of one or more sessions, every session has a sessionId, and sessions can have tracks, participants, tags and URI's associated with them and so forth. However, the details of the SIP-level interaction with MediaSense diverge somewhat depending on whether the call is controlled by a CUBE or by a Unified Communications Manager. This leads to some differences in the metadata that MediaSense clients observe and in the real time events they receive.

- The first difference is in the topology: with Unified Communications Manager, calls are forked by one of the endpoints. With CUBE, calls are forked by a network element (ISRG2) through which the call passes. With CUBE, the RTP media and the SIP dialog come from the same device; with Unified Communications Manager, they are different devices. Therefore the way participants are identified and in the way they are associated with media tracks is different.
- More substantial differences in the metadata come from mid-call activities. For example, hold and resume trigger a new session for Unified Communications Manager calls, but they insert track level tags for CUBE calls. Transfer of a forking phone terminates a recording session on Communications Manager, but such an action is not possible with CUBE calls.
- Conference detection varies considerably too. With Unified Communications Manager, this action appears as a transfer to a conference bridge with updated participants and a metadata flag identifying it as a conference. With CUBE, the action also appears as a transfer, but the metadata flag is not supported and the participant data is erratic and unreliable.
- Another difference is with respect to call correlation. Unified Communications Manager and CUBE use different mechanisms for identifying calls.

Simple application clients can be agnostic to call controller type, but more sophisticated clients will usually need to know whether a call was managed by a CUBE or by a Unified Communications Manager.

The Cisco MediaSense Developers Guide contains a full description of the differences between CUBE and Unified Communications Manager deployments.



CHAPTER 5

Disk space management

As on any recording device, disk space is a critical resource. MediaSense provides a number of features designed to meet various, sometimes conflicting, space management needs.

MediaSense has two different disk partitions for storing media files: one for recorded media, and one for uploaded media. The former is used for continuous recording and playback of conversations; the latter (which is usually much smaller) is used for ViQ, VoD and VoH. Video voice-mail greetings are stored with recorded media. Both partitions are assigned their minimum size automatically at installation and both may be assigned additional space when needed.

The uploaded media partition must be managed entirely by the administrator. It is the administrator's job to keep track of the amount of space in use and available and manually deleting unneeded videos or adding new storage as required. Be aware that the amount of storage required for a single uploaded .mp4 file does not equal the size of that file. MediaSense automatically converts it upon upload into multiple formats and distributes it to all nodes in the cluster (so that the right video is ready to be played on the right node when a caller requires it). The only way to determine how much space a given video will occupy is to upload that video, wait for it to become ready, and then check the Media Partition Management page to see how much space was used.

The recorded media partition is much more dynamic and can be managed either automatically by MediaSense, or explicitly by a MediaSense API client application.

Recorded media partition

At a high level, two space management operating modes are available:

- Recording priority—for customers who would rather lose an old recording than miss a new one.
- Retention priority—for customers who would rather miss a new recording than lose an old one.

In recording priority mode, MediaSense automatically prunes recordings that age beyond a configurable number of days or when the percentage of available disk space falls to dangerous levels.

Retention priority mode focuses on media retention and MediaSense does not automatically prune recordings for any reason.

In either mode, MediaSense stops accepting new calls when necessary to protect the space remaining for calls that are currently in progress. Affected calls are automatically redirected to another MediaSense recording server (if one is available).

Retention priority behavior

- No automatic pruning takes place.
- When a server enters the warning condition (75%), an alarm is raised.
- When a server enters the critical condition (90%), it redirects new calls.
- When a server enters the emergency condition (99%), it drops active recordings.
- When a server exits a critical condition (drops below 87%), it starts taking new calls.

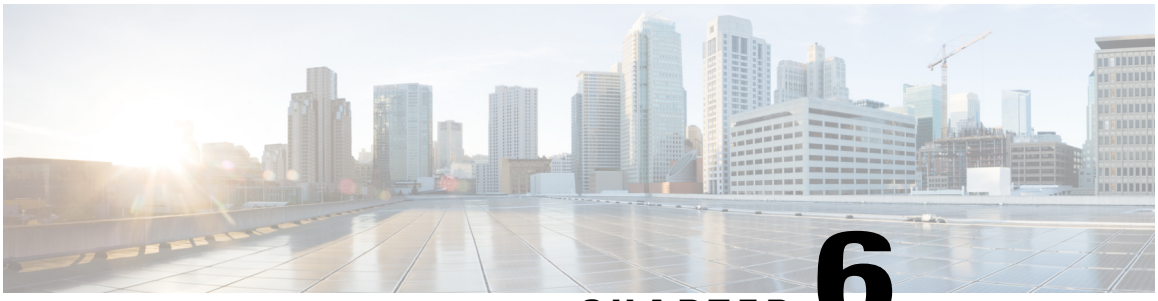
Recording priority behavior

- Automatic age-based pruning is in effect; recordings older than a configurable number of days are automatically pruned.
- When a server enters the warning condition (75%), an alarm is raised.
- When a server reaches the critical condition (90%), older recordings (even if younger than the age threshold) are pruned to make room for new ones.
- When a server enters the emergency condition (99%), it redirects new calls and drops ongoing recordings.
- When a server exits the an emergency condition (drops below 97%), it starts taking new calls.

Any automatic pruning applies only to raw recording files. An administrative option determines whether MediaSense should automatically delete any.mp4 recordings that were created using the deprecated convertSession API, as well as any metadata associated with pruned recordings. If this option is not enabled, an API client must take responsibility for deleting them (.mp4 and .wav files that are created dynamically by the mp4url or wavUrl mechanisms do get cleaned up automatically by the system, but the API client need not concern itself with them).

Clients also have the option of managing disk usage directly. MediaSense takes progressively more aggressive action when storage levels reach more dangerous levels, but as each stage is entered or exited, it publishes an event to subscribed clients. These events inform the client when space management actions are necessary. The MediaSense API offers a number of options to use for deleting recordings—including an option to issue a customized bulk delete operation that is then carried out without client involvement.

The capability to explicitly delete old recorded sessions is not limited to automatic operations performed by a server-based client. A customer can take a completely manual approach, for example, designing a web page that fetches and displays appropriate meta-information about older recordings and allowing an administrator to selectively delete those that he or she considers to be expendable. Such a web page would use the same API that the server-based client would use.



CHAPTER 6

System resiliency and overload throttling

MediaSense keeps track of a number of metrics and statistics about its own performance and raises alarms when certain thresholds are exceeded. The system also protects itself by rejecting requests that would cause it to exceed its critical capacity limits. When the node is at capacity, new recordings are redirected to other nodes (if available) or rejected and lost.

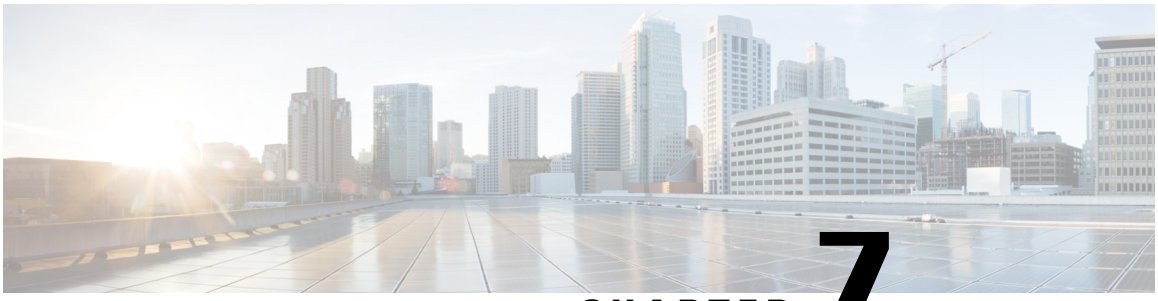
Since recording is always considered to be the highest priority operation, MediaSense reserves a certain amount of capacity specifically for that purpose, electing to reject media output requests while still continuing to accept new recording requests. Media output requests (such as live monitoring, playback, raw download and .mp4 or .wav conversion) result in 503 responses when the node is at capacity.

The relative weight of various media is also considered for overload throttling. For example, video takes significantly more capacity than audio.

Note however, that these are overload protection mechanisms only; they are not intended to enforce licensed or rated capacity. They reflect the levels at which the product has been tested and they exist so that MediaSense nodes can protect themselves and offer graceful service degradation in case of severe overuse. **It is still the customer's responsibility to engineer his or her deployment such that the overall rated node and cluster capacities are not exceeded.**

MediaSense also protects itself with respect to media storage capacity. It raises alarms, redirects new calls to other nodes (if available), prunes older recordings to recover space (if permitted), and even drops existing calls (as a last resort) in order to maintain the integrity of existing recordings.

The Real Time Monitoring Tool (RTMT) provides a great deal of statistical information about use levels and throttling activities for each node.



CHAPTER 7

MediaSense-specific deployment models

This sections discusses the various options for deploying MediaSense servers and virtual machines.

Server models supported

MediaSense can only be deployed on a VMware hypervisor, which must be running on a

- Cisco B-series or C-series server with fiber channel-attached SAN storage (for at least the first two disks) or with directly attached hard disk drives
- or on a UCS-E module running within a Cisco ISR-G2 router
- or on selected Hewlett Packard (HP) or IBM servers (see *Compatibility matrix* below for versions and model numbers).

When ordering C-series servers, be sure to include either the battery backup or Super Cap RAID controller option. If one of these is not present or not operational, the write cache is disabled on these controllers. When the write cache is disabled, write throughput is significantly reduced. (See *Compatibility matrix* below for detailed disk requirements.)

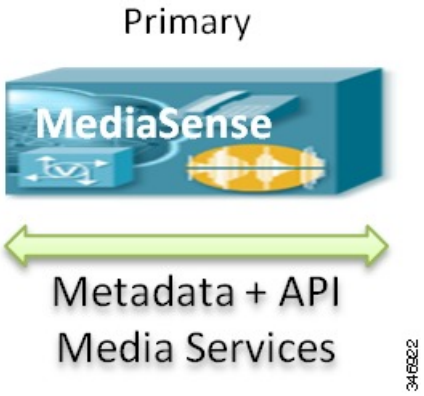
The primary and secondary servers must be based on identical hardware, or at least have identical specifications in terms of CPU speed and disk I/O throughput. They must also be using the same version of VMware ESXi. Any asymmetry causes accumulating database latency in one server or the other. Expansion servers do not need to be running on the identical hardware.

Server types

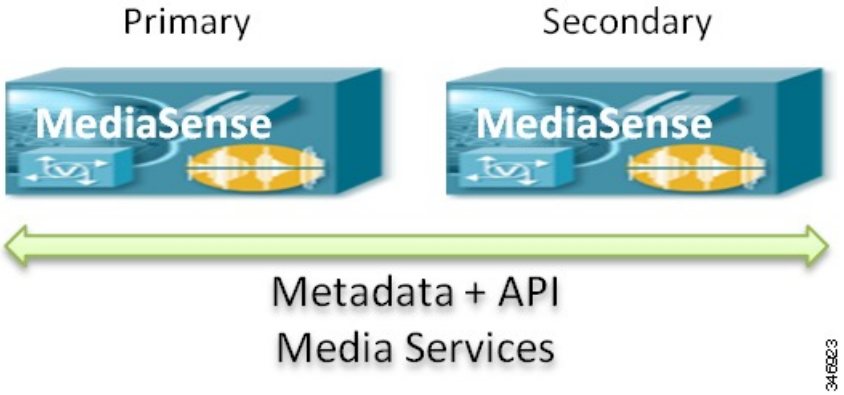
MediaSense is deployed on up to five rack-mounted servers or up to two UCS-E modules, depending on the capacity and degree of redundancy required. (in this context, "server" refers to a virtual machine, not necessarily a physical machine). There are three types of servers:

- Primary (required): Supports all database operations as well as media operations.
- Secondary (optional): Supports all database operations as well as media operations. Provides high-availability for the database.
- Expansion (optional): Provides additional capacity for media operations, but not for database operations. Expansion servers are only used in 7-vCPU deployments, and are never used in UCS-E module deployments.

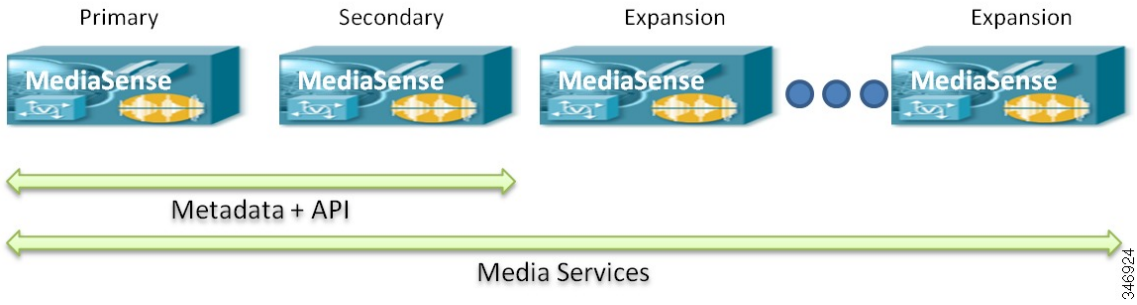
The following diagram depicts a primary-only deployment model:



Customers who require database redundancy can deploy a secondary server, as shown below:



If additional recording capacity is required, expansion servers are deployed, as shown below:



Note Expansion servers are not supported in deployments which do not use the full 7-vCPU template.

All servers (including UCS-E servers) run the same installed software; they differ only in function and capacity. The primary server is always the first server to be installed and is identified as such during the installation process. Secondary and expansion servers are identified during the initial web-based setup process for those nodes (after installation is complete).

Recordings are always stored on the disks which are attached to the server which initially captured the media.

UCS-E-based two-server clusters may be deployed with both blades in the same ISRG2 router or with one blade in each of two ISRG2 routers. The latter is typically preferred from a fault isolation perspective but is not required. A MediaSense cluster must be UCS-E-based or rack-mount server based. It cannot be made up of a combination of the two.

Grow your system

There are two reasons for expanding system capacity: to be able to handle more simultaneous calls, and to increase the retention period of your recordings.

If your goal is to handle more calls, then you can simply add nodes to your cluster. Each nodes adds both simultaneous activity capacity – the ability to perform more parallel recording, monitoring, download and playback activities – and storage capacity. Servers may be added to a cluster at any time, but there is no ability to remove servers from a cluster.

Once your cluster has reached its maximum size (5 nodes for 7 vCPU systems, 2 nodes for everything else), your only option is to add a new cluster. If you do, you must arrange your clusters with approximately equal numbers of nodes. If that is not possible, then you should at least arrange the call distribution such that an approximately equal number of calls is directed to each cluster, possibly leaving the larger cluster underutilized. For more information, see Very large deployments.

There is no capability to remove a node from a cluster once it has been added.

If your intention is to achieve a longer retention period for your recordings, there are two options. You can add nodes, as described above, or you can provision more storage space per node. Each node can handle up to 12 TB of recording storage, as long as the underlying hardware can support it. UCS-B/C servers with SAN can do so easily, but servers using direct attached storage (DAS) and low-end UCS-E servers are more limited.

As with nodes in a cluster, there is no capability to remove storage capacity from a node once it has been installed.

Very large deployments

Customers who require capacity that exceeds that of a single MediaSense cluster must deploy multiple independent MediaSense clusters. In such deployments, there is absolutely no interaction across clusters; no cluster is aware of the others. Load balancing of incoming calls does not extend across clusters. One cluster might reach capacity and block subsequent recording requests, even if another cluster has plenty of remaining room.

API clients need to be designed such that they can access all of the deployed clusters independently. They can issue queries and subscribe to events from multiple clusters simultaneously, and even specify that multiple clusters should send their events to the same SWS URL. This permits a single server application to receive all events from multiple clusters. Some MediaSense partner applications already have this capability. MediaSense's built-in Search and Play application, however, does not.

The following list is a summary of whether multiple call controllers can share one MediaSense cluster and whether one call controller can share multiple MediaSense clusters.

- Multiple Unified CM clusters to one MediaSense cluster: Supported.
- Multiple CUBE devices to one MediaSense cluster: Supported.
- One Unified CM cluster to multiple MediaSense clusters: Only supported with partitioned phones.
- One CUBE to multiple MediaSense clusters: Supported.

Calls forked by Unified CM phones

For Unified Communications Manager phone recording, phones must be partitioned among MediaSense clusters such that any given phone will be recorded by one specific MediaSense cluster only; its recordings will never be captured by a server in another cluster. (But there is an option for full cluster failover, such as in case of a site failure. See High availability.)

In some deployments there is a need for multiple Unified CM clusters, possibly tied together in an SME network, to share one or a small number of MediaSense clusters. In this arrangement, there is a chance that two calls from different Unified CM clusters will carry the same pair of xRefCi values when they reach MediaSense. This would cause those two calls to be incorrectly recorded or not recorded at all. (The statistical probability of this sort of collision is incredibly small and need not be considered.) This arrangement, therefore, is fully supported.

If your application uses the startRecord API function that asks MediaSense to make an outbound call to a specified phone number and begin recording it, only a single Unified Communications Manager node may be configured as the call controller for this type of outbound call.

Calls forked by CUBE

For CUBE recording, the situation is more flexible. One or more CUBE systems may direct their forked media to one or more MediaSense clusters. There is no chance of a call identifier collision because MediaSense uses CUBE's Cisco-GUID for this purpose, and that GUID is globally unique.

Since MediaSense clusters do not share the load with each other, CUBE must distribute the load with its recording invitations. This can be accomplished within each CUBE by configuring multiple recording dial peers to different clusters so that each one is selected at random for each new call. Alternatively, each CUBE

can be configured with a preference for one particular MediaSense cluster, and other mechanisms (such as PSTN percentage allocation) can be used to distribute the calls among different CUBE devices.

If your goal is to provide failover among MediaSense clusters rather than load balancing, see High availability.

Virtual machine configuration

Cisco provides an OVA virtual machine template in which the minimum size storage partitions and the required CPU and memory reservations are built in. VMWare ESXi enforces these minimums and ensures that other VMs do not compete with MediaSense for these resources. However, ESXi does not enforce disk throughput minimums. Therefore, you must still ensure that your disks are engineered such that they can provide the specified IOPS and bandwidth to each MediaSense VM.

The Cisco-provided OVA template contains a dropdown list that allows you to select one of the supported VMware virtual machine template options in each release for MediaSense servers. These template options specify (among other things) the memory, CPU, and disk footprints for a virtual machine. For each virtual machine you install, you must select the appropriate template option on which to begin your installation. You are required to you use the Cisco-provided templates in any production deployment. For low volume lab use, it is possible to deploy on lesser virtual equipment, but the system will heavily constrain its own capacity and display warning banners that you are running on unsupported hardware.

The following table summarizes the template options provided.

Template option	vCPUs	Memory	Disk	CPU reservation	Maximum total recording space supported	Notes
Primary or secondary node	2	6 GB	80 GB for OS 80 GB for DB 210 GB for media	2200 MHz	420 GB across two nodes. No expansion nodes supported. Users can add additional media disks.	2,4
Primary or secondary node	4	8 GB	80 GB for OS 600 GB for DB 210 GB for media	3200 MHz	24 TB across two nodes. No expansion nodes supported. Minimum 210 GB per node provisioned by default Users can add additional media disks.	UCCX integration should use this template 2,3,4

Template option	vCPUs	Memory	Disk	CPU reservation	Maximum total recording space supported	Notes
Expansion node	7	16 GB	80 GB for OS 80 GB for DB	10000 MHz	12 Terabytes per expansion node. Minimum 210 GB per node provisioned by default. Users can add additional media disks.	1,2,3
Primary/secondary node	7	16 GB	80 GB for OS 600 GB for DB 210 GB for media	15000 MHz	12 Terabytes per primary or secondary node. Minimum 210 GB per node provisioned by default. Users can add additional media disks.	2,3

Notes:

1. All rack-mount expansion servers must use the expansion template option.
2. All disks must be "thick" provisioned. Thin provisioning is not supported.
3. The primary, secondary and expansion OVA template options provision the minimum 210 GB by default. Additional space may be added before or after MediaSense software installation. Once provisioned, recording space may never be reduced. The total amount of media storage across all nodes may not exceed 60 Terabytes on 5-node clusters or 24 Terabytes on 2-node clusters.
4. The primary and secondary node 2 vCPU and 4 vCPU templates are suitable for UCS-E blade deployments, although they can also be used on larger systems. Most supported UCS-E blades have more physical disk space available than the VM template allocates; the unused space may be used for recorded media or uploaded media.

Media storage alternatives

On rack-mount servers, two storage alternatives for recorded data are currently supported:

- Physically attached disks; or
- Fiber channel-attached SAN

Note Network Attached Storage (NAS) is not supported in any MediaSense configuration and SAN storage is not supported on UCS-E configurations.

Depending on the hardware model and options purchased, any single node can offer up to 12TB of storage—a maximum of 60TB of storage across five servers. It is not necessary for all servers to be configured with the same number or type of virtual disks. (See the Compatibility matrix section for detailed specifications and other storage configuration requirements.)

RAID configurations

This section is applicable to UCS C-series servers only.

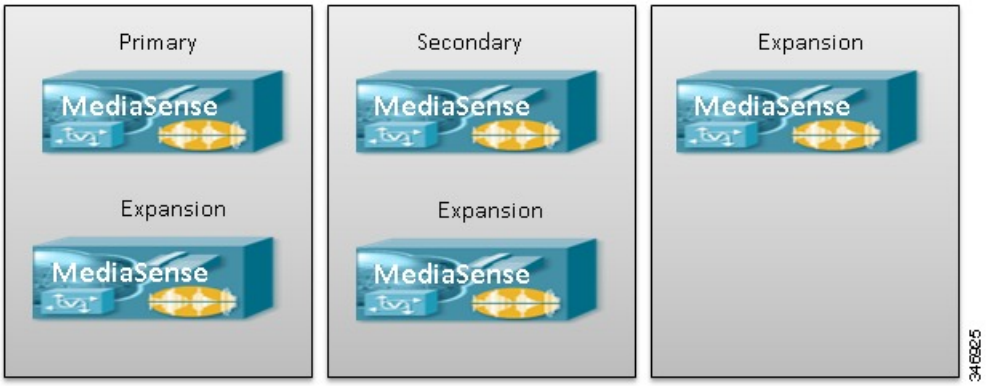
MediaSense must be configured with RAID-10 for the database and OS disks and either RAID-10 or RAID-5 for media storage. Using RAID-5 results in hardware savings. It is slower, but fast enough for media storage. All of the TRC configurations for UCS C-series servers include an internal SD card which is large enough to house the ESXi hypervisor. Therefore, Cisco supports installation of ESXi on the SD card and having the MediaSense application installed on the remaining disk drives.

That RAID-10 group would have to hold the ESXi hypervisor as well as the MediaSense application, which is not generally a recommended practice. Fortunately, all the TRC configurations for UCS C-series servers include an internal SD card which is large enough to house ESXi. Cisco therefore recommends that ESXi be installed on the SD card and the MediaSense application be installed on the remaining disk drives.

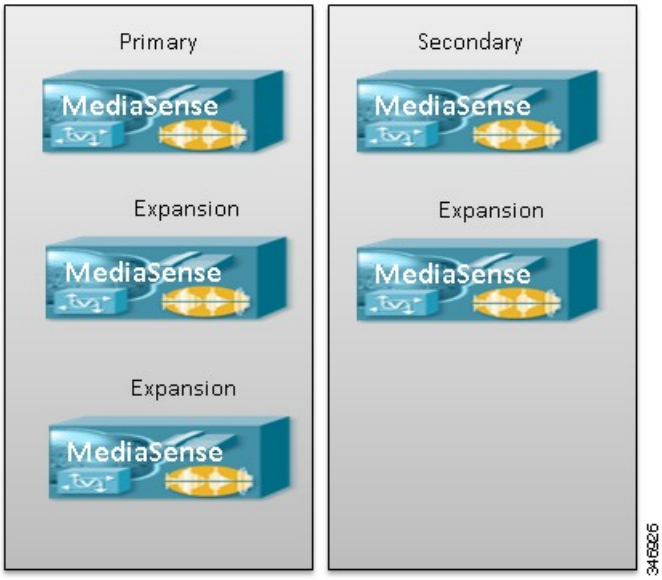
Deploy multiple MediaSense virtual machines per server

As of Release 9.0(1), MediaSense is no longer required to be the only virtual machine running on a physical server. Other applications can share the server, as long as MediaSense is able to reserve the minimum resources it requires.

You can also deploy multiple MediaSense VMs on a single host. When doing so however, ensure that the primary and secondary nodes do not reside on the same physical host. For example, if your servers can support two MediaSense VMs, then you might lay out a 5-node cluster in this way:



If your servers can support three MediaSense VMs, then you might lay them out as follows:



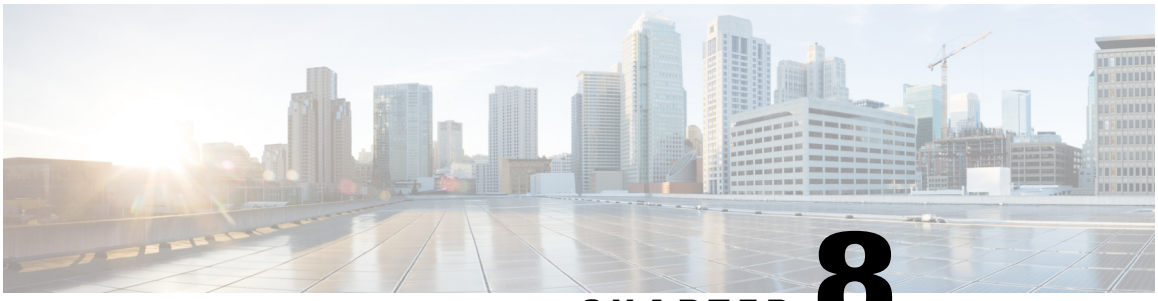
You can determine how many MediaSense VMs a particular server model will support by referring to the UC Virtualization Wiki and use the number of physical CPU cores as a guide. Models with 8 or more physical cores can support 1 MediaSense VM; models with 14 or more physical cores can support 2 MediaSense VMs, and models with 20 or more physical cores can support 3 MediaSense VMs.

Geographical specifications

All MediaSense servers within a cluster must be in a single campus network. A campus network is defined as a network in which the maximum round-trip delay between any pair of MediaSense servers is less than 2 milliseconds. (Some Metropolitan Area Networks (MANs) may fit this definition as well.)

Other solution components, however, may connect to the MediaSense cluster over a WAN, with certain caveats:

- In Unified Communications Manager deployments, media forking phones may be connected to MediaSense over a WAN.
- SIP Trunks from Unified Communications Manager may also be routed over a WAN to MediaSense, but calls may evidence additional clipping at the beginning of recordings due to the increased round trip delay.
- The connection between CUBE and MediaSense may be routed over a WAN with the same warning—affected calls may evidence additional clipping at the beginning of recordings due to the increased round trip delay.
- The AXL connection between Unified Communications Manager and MediaSense may be routed over a WAN, but API and administrator sign-in times may be delayed.
- From a high availability standpoint, API sign-in has a dependency on the AXL link. If that link traverses a WAN which is unstable, clients may have trouble signing in to the API service or performing media output requests such as live monitoring, playback, and recording session download. This applies to remote branch deployments as well as centralized deployments, and to CUBE deployments as well as Unified CM deployments.

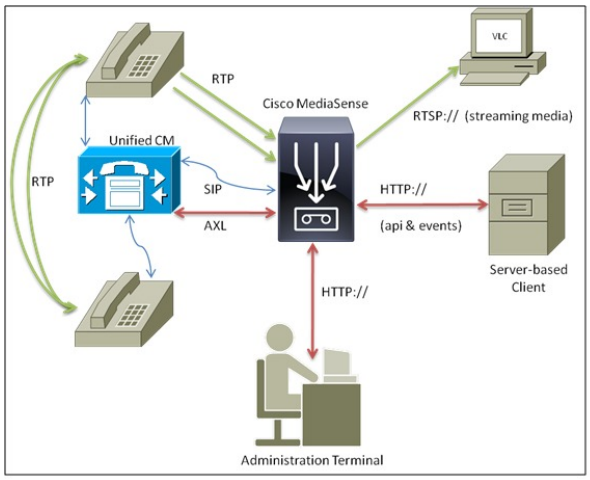


CHAPTER 8

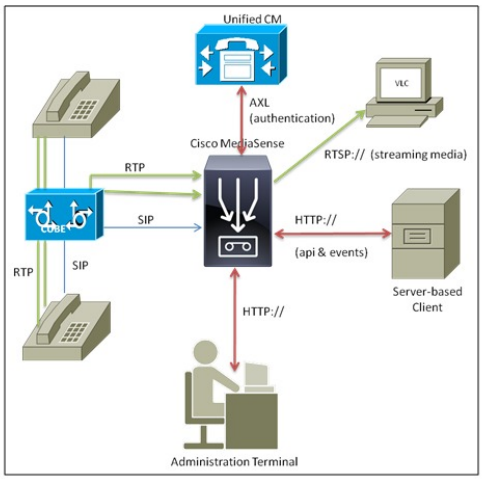
Solution environment

Solution overview and call flows

The following diagrams depict the MediaSense solution environment for Unified Communications Manager deployments and for CUBE deployments:



UCM Solution Topology



CUBE Solution Topology

346911

Though these diagrams each show only one MediaSense server and one Unified Communications Manager server or CUBE, each should be considered as a cluster of such devices. That is, one cluster of MediaSense servers interacts with one cluster of Unified Communications Manager servers or with one or more CUBE devices.

For Unified Communications Manager deployments, there is no concept of a hierarchy of recording servers. SIP Trunks should be configured to point to all MediaSense servers.

For CUBE deployments, recording dial peers should be configured to point to one or two of the MediaSense servers (preferably avoiding the primary and secondary). The High availability section discusses this in more detail.

UCS-E deployments are built with exactly the same topology. Physically, a UCS-E module is a blade inserted into a router rather than a separate rack-mounted server; but logically it functions no differently within the solution environment than does a rack-mounted server. A UCS-E-based MediaSense cluster can even record calls that are forked from Unified Communications Manager phones.

Notice that the CUBE solution topology includes a Unified Communications Manager device. This is used only for authentication purposes and has no role in call flow.

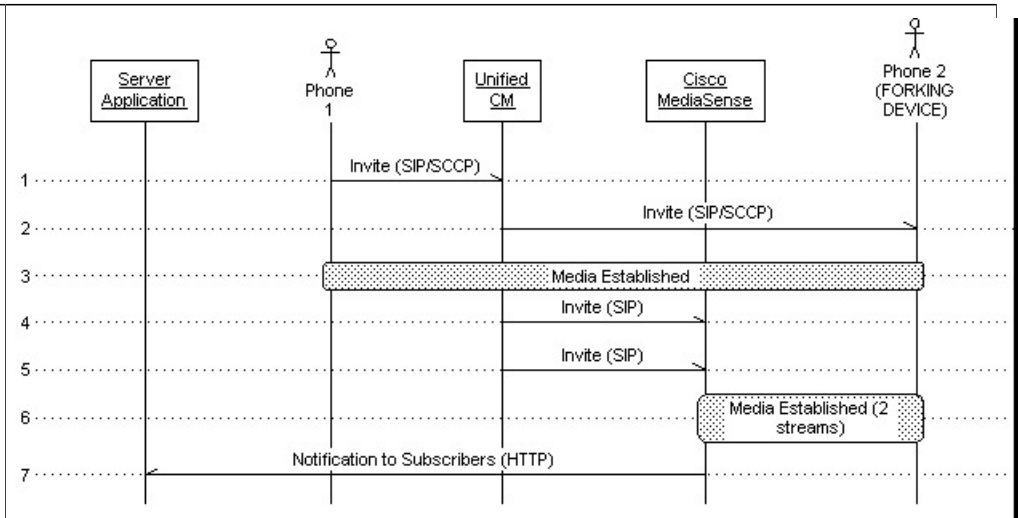
General flow - Unified Communications Manager calls

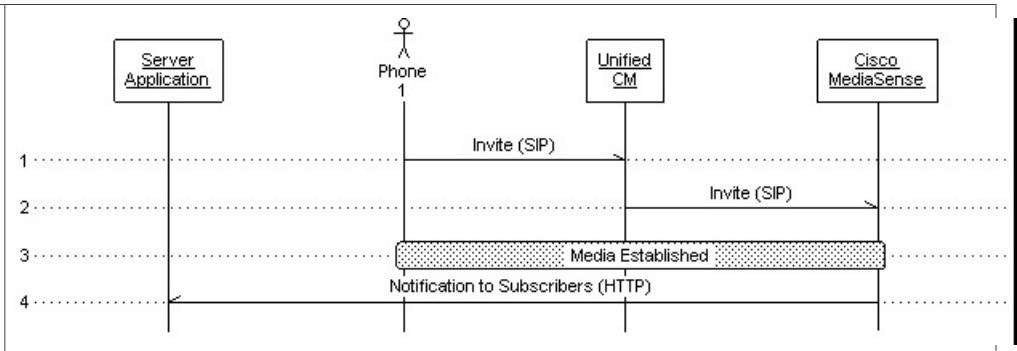
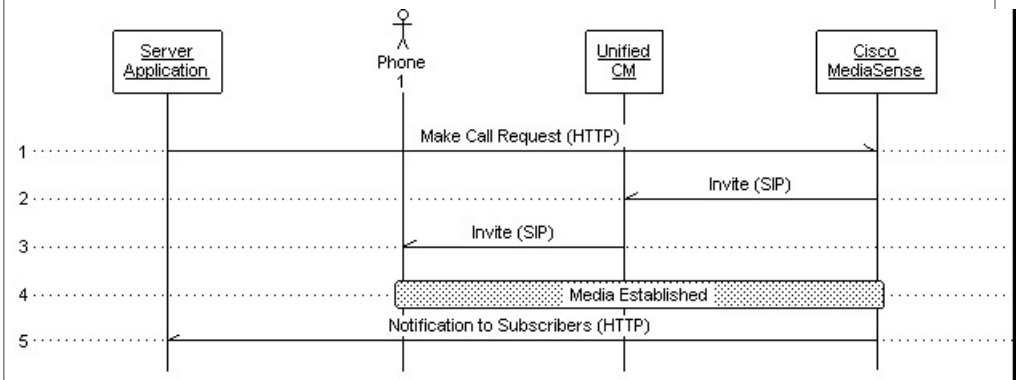
For compliance recording applications, call recordings are initiated via a pair of SIP invitations from Unified Communications Manager to MediaSense after the initial call has been established between two parties. Unified Communications Manager is involved in the call setup but media flows to MediaSense from one of the phones—not from Unified Communications Manager.

Inbound blog recordings are initiated in a similar way. A SIP invitation is sent from Unified Communications Manager to MediaSense. Outbound blog recordings are initiated with an API request ("startRecording") to MediaSense, which triggers an outbound SIP invitation from MediaSense to Unified Communications Manager. In all cases, the processing of the invitation results in one or more RTP media streams being established between the phone being recorded and MediaSense. These call flows are depicted in the following figures.

Note These figures are for illustration purposes only and do not show the detailed message flow.

Compliance Recording for Unified CM Calls



Direct Inbound Recording for Unified CM Calls	
Direct Outbound Recording for Unified CM Calls	

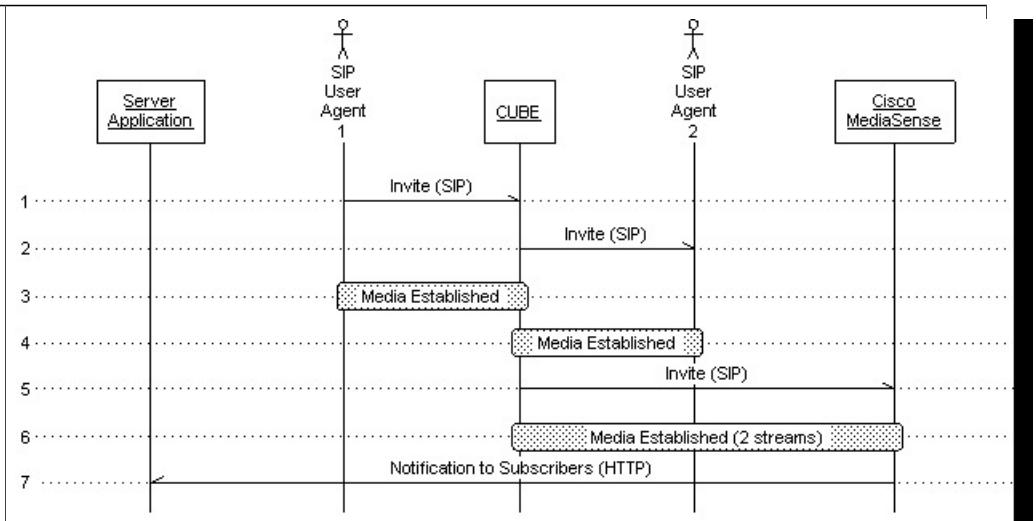
General flow - CUBE calls

In CUBE recording, applications have similar flows, but there are important differences. Call recordings are initiated with a single SIP invitation from CUBE to MediaSense containing two "m" lines, as opposed to two separate invitations that each contain one "m" line. As with Unified Communications Manager, the invitation is sent only after the initial call has been established between two parties. However, the media that flows to MediaSense comes from CUBE, not from one of the endpoints.

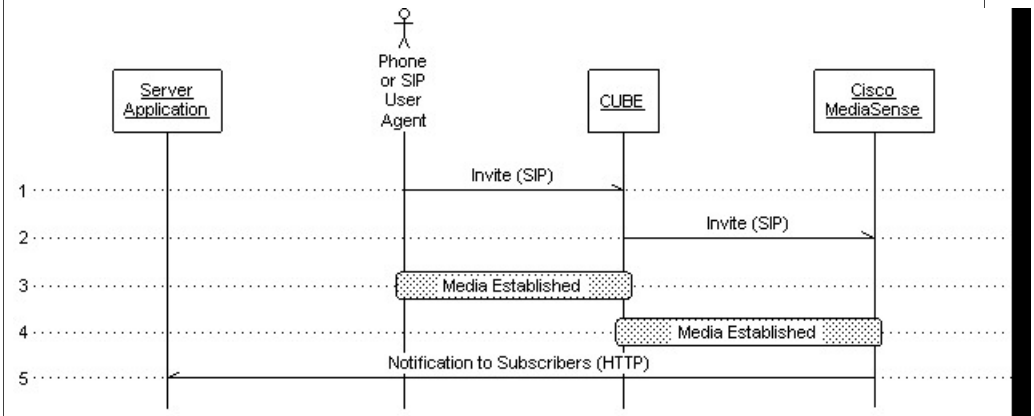
Inbound blog recordings are initiated by directly dialing a call from an endpoint, through CUBE, to MediaSense. In all cases, the processing of the invitation to MediaSense results in one or more RTP media streams being established between the CUBE and MediaSense. These call flows are depicted in the following figures.

Note These figures are for illustration purposes only and do not show the detailed message flow. Also, outbound blog recordings are not supported with CUBE deployments.

Compliance Recording for CUBE Calls



Direct Inbound Recording for CUBE Calls

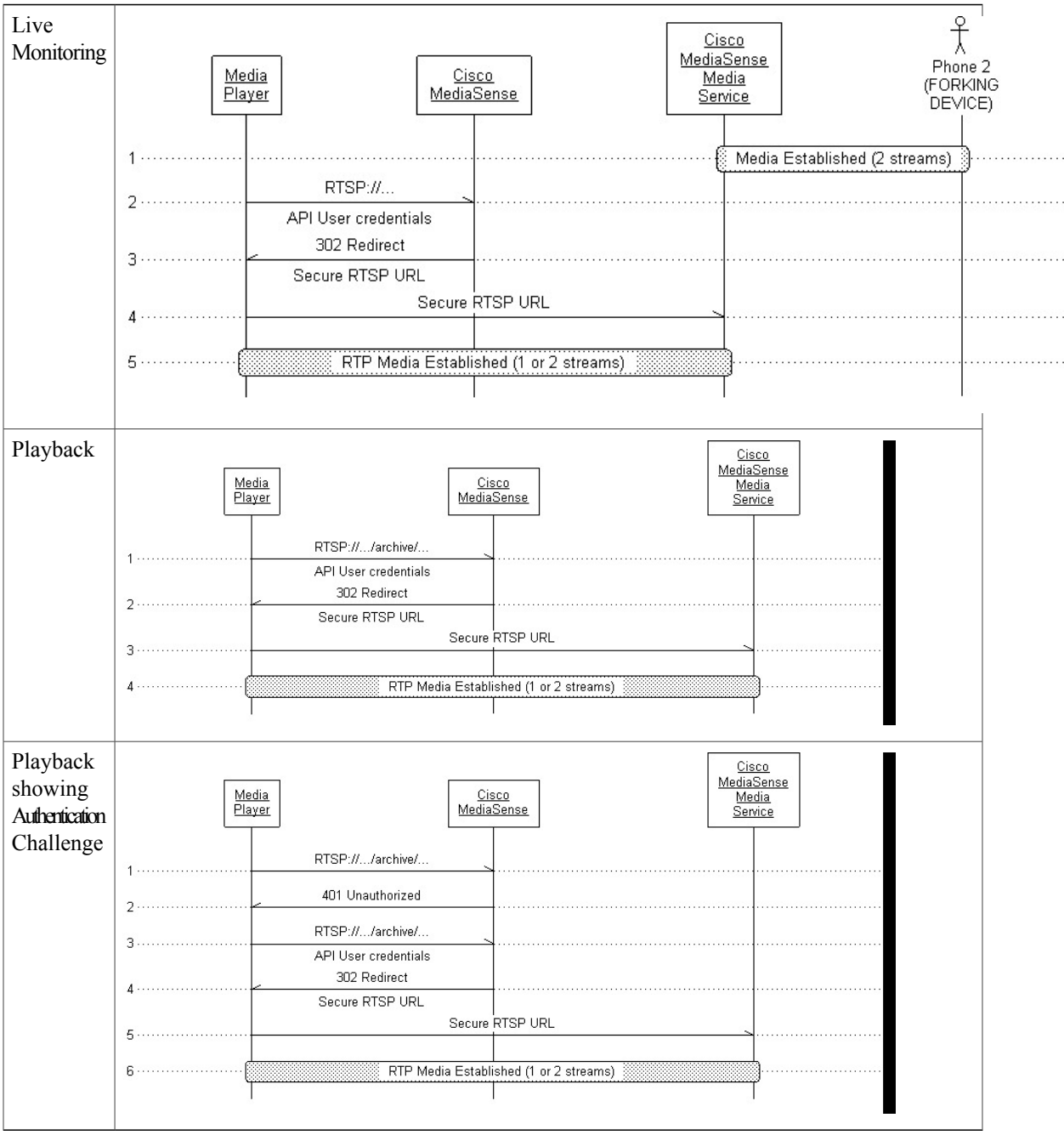


General flow - streaming media

Live monitoring happens when a workstation running a streaming media player sends an RTSP:// URI to MediaSense specifying an active media address and an RTP media stream is established between MediaSense and the player. This stream is actually a copy of one of the streams that MediaSense is receiving from the phone; the media does not come from the disk.

Playback is initiated when a workstation running a streaming media player sends an RTSP:// URI to MediaSense specifying an "archive" media address. The resulting media stream between MediaSense and the player is read from the disk.

Live monitoring and playback call flows are illustrated below (showing how authentication takes place, but not showing the detailed message flow). The MediaSense Media Service is the software component within each node that is responsible for handling streaming media.



The MediaSense API is accessed from either a server-based or a browser-based client. Server-based clients may subscribe for asynchronous events as well.

Solution-level deployment models

This section summarizes the many ways in which MediaSense can be deployed as part of a solution.

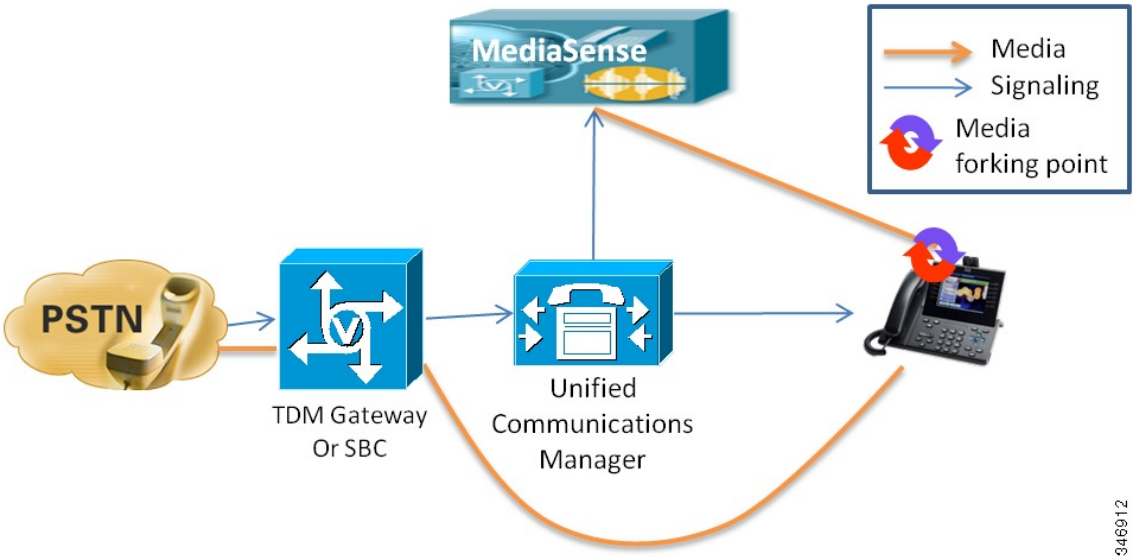
Unified Communications Manager deployments

These deployment models cover scenarios in which Cisco IP phones are configured for media forking. Two versions are covered:

- Basic Unified Communications Manager deployment - internal-to-external
- Basic Unified Communications Manager deployment - internal-to-internal

From the perspective of MediaSense, there is actually no difference between the two basic Unified Communications Manager versions. In both cases, media forked by a phone is sent to the recording device where the forked streams are captured. They are distinguished here because there is a significant difference in their behavior at the solution level.

Unified Communications Manager deployment - internal-to-external

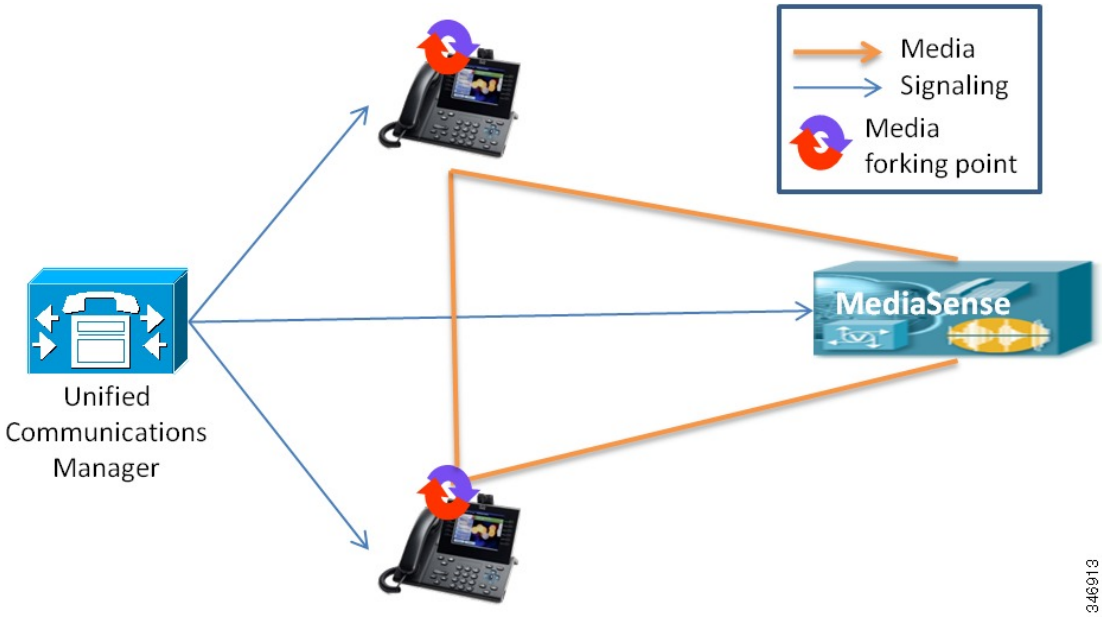


The preceding diagram shows a basic Unified Communications Manager deployment where calls with parties who are outside the enterprise are recorded. This applies to both inbound and outbound calls, as long as the inside phone is configured with an appropriate recording profile.

Once the connection is established from a signaling perspective, media flows directly from the forking phone to the recording server.

If the call is transferred away from this phone, the recording session ends. Only if the phone which takes up the call is configured for recording will the next segment of the call be captured.

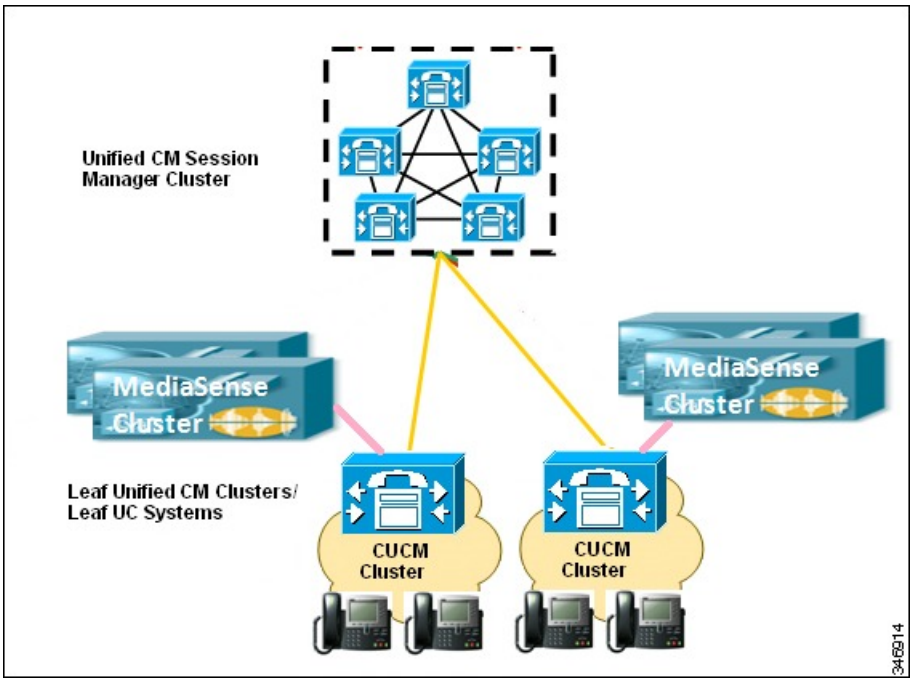
Unified Communications Manager deployment - internal-to-internal



This diagram shows a basic Unified Communications Manager deployment where calls are with parties who are inside the enterprise. One of the phones must be configured for recording. If both phones are configured for recording, then two separate recording sessions are captured.

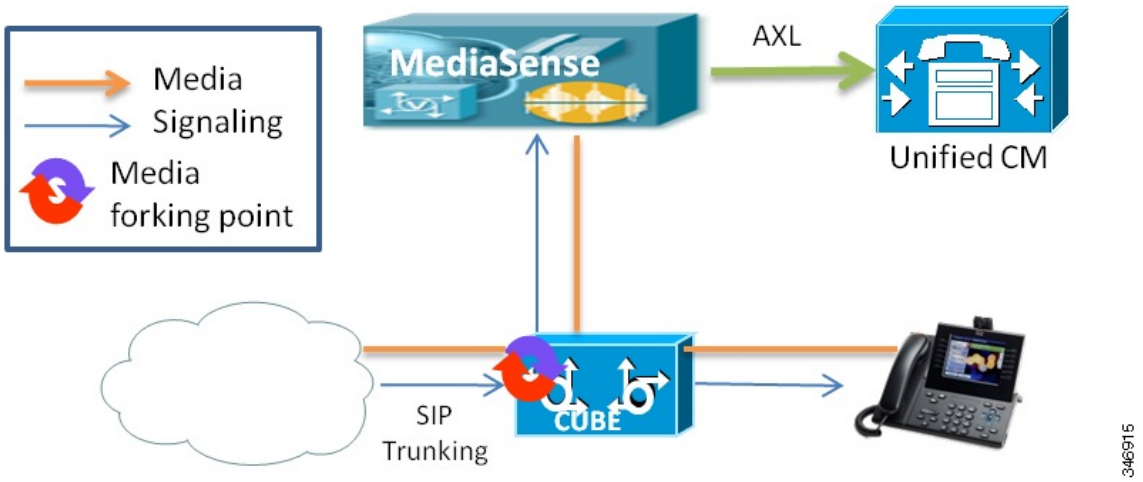
Unified SME deployment

At this time, MediaSense does *not* support SME deployments in the general sense. In an SME environment, all the phones that are to be recorded by a specific MediaSense cluster must be part of the same SME leaf cluster. If phones from different leaf clusters need to be recorded, then separate and independent MediaSense clusters must be deployed, as shown in this diagram.



The preceding diagram demonstrates how MediaSense clusters must be connected to SME leaf clusters, not to the SME manager cluster. The diagram also shows the leaf clusters connecting to separate MediaSense clusters. That is a supported arrangement, but it is also acceptable for them to share one or more MediaSense clusters.

Basic CUBE deployment



The preceding diagram shows a very basic CUBE deployment where calls arrive on a SIP Trunk from the PSTN and are connected to a SIP phone inside the enterprise. The media forking is performed by the CUBE device using a recorder profile configuration that is attached to one or more dial peers.

When a call passes through CUBE (or any Cisco router for that matter), it matches two dial peers - one at the point where the call enters the CUBE, and one at the point where it exits. From the CUBE system perspective,

these are known as the inbound and outbound dial peers. These terms are relative to the direction of the call. On an inbound call, the inbound dial peer is the one that represents the outside of the enterprise and the outbound dial peer represents the inside of the enterprise. The assignment is reversed for an outbound call. In this document, we use the terms inside and outside dial peers to represent the inside and the outside of the enterprise respectively.

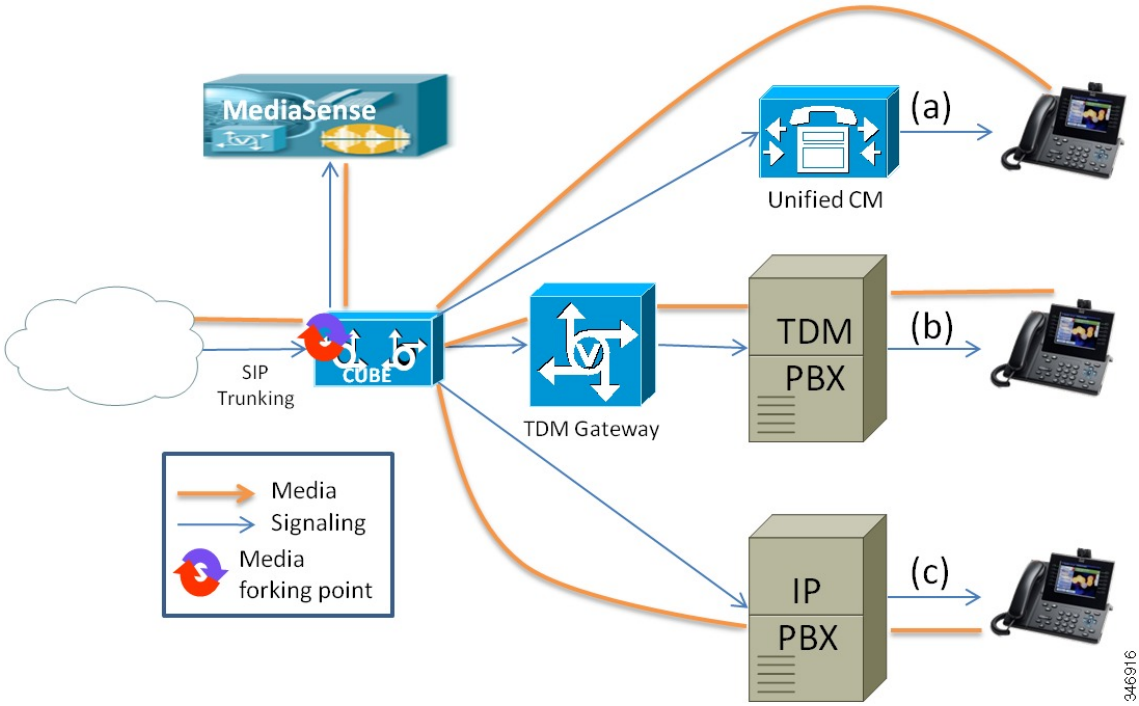
Although there are a few exceptions, it is a best practice is to apply the recording profile to the outside dial peer—the inbound dial peer for inbound calls and the outbound dial peer for outbound calls. This is because the external leg of the call is typically quite stable, whereas the internal leg is often subject to complex call manipulations including various kinds of consults, conferences, and transfers. If any of those operations cause the CUBE to trigger a new dial peer match, the recording session may be terminated prematurely. (If such an operation causes the prevailing codec to be changed, the recording session is terminated and a new one is initiated.)

This diagram also shows a Unified Communications Manager component. Though currently required for CUBE deployments, Unified Communications Manager does not perform any call signaling, media, or record keeping. A single Unified Communications Manager server is required to manage and authenticate MediaSense API users. It can be any existing or specially installed Unified Communications Manager server on Release 8.5(1) or later. Ideally, the server selected should be one that is not itself loaded with calls.

The Unified Communications Manager server is omitted from the remaining CUBE deployment model diagrams since it plays no part in call handling.

The basic CUBE deployment is unlikely to ever be used in a production environment. More typically, a Unified Communications Manager, other Private Branch Exchange (PBX), or Automatic Call distributor (ACD) would be attached to the internal side of the CUBE and phones would be attached to that rather than to the CUBE directly. However, all CUBE deployments contain this configuration at their core. From the strict perspective of CUBE and MediaSense, all the other models are no different from this one.

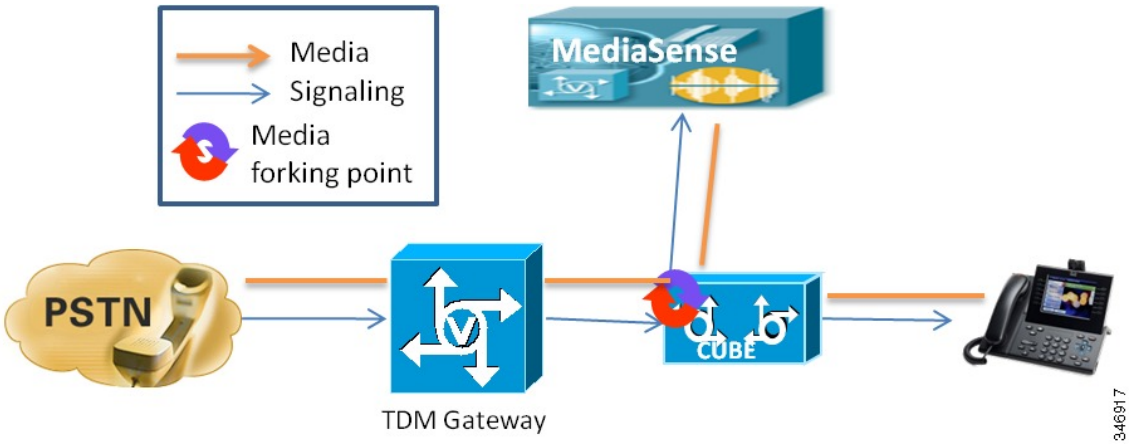
Basic CUBE deployment with various PBXs



One of the great advantages of using CUBE to fork media is its ability to capture the entire conversation from the caller perspective, no matter where the call goes inside the enterprise. This includes contact center agents, non-contact center personnel, IVR systems, and even users on non-Cisco ACD and PBX systems.

The preceding diagram shows three ways that MediaSense and CUBE may be deployed in a heterogeneous enterprise environment. Any given call might experience one or a combination of these flows and the entire caller experience will be recorded. Additional combinations are possible as well; for example a call may be handled by an IP-based or TDM-based IVR system.

CUBE deployment variation using TDM ingress



In order to fork media, CUBE must be dealing with a SIP to SIP call. If calls are arriving by TDM, then a separate TDM gateway is provisioned as shown in the diagram above. Forking is then be configured as usual on the outside dial peer of the CUBE.

If your application is designed to transmit DTMF signals to the PSTN, such as to perform PSTN-controlled transfers (also known as *8 Transfer Connect), then you must ensure that both the CUBE and the TDM gateway are configured to use the same method for DTMF signaling. You can do so by adding the same "dtmf-relay" configuration to the connecting dial peers in both devices. Relay type "rtp-nte" is the most standard, preferred method. The dial peer going to CVP should also be configured with rtp-nte.

CUBE deployments with CVP

When CUBE is connect to Customer Voice Portal (CVP), MediaSense can be used to record calls in a contact center. The following subsections describe these models:

- CUBE deployments with Unified CVP - centralized, SIP trunks, no survivability
- CUBE deployments with Unified CVP - centralized, SIP trunks, with survivability
- CUBE deployments with Unified CVP - centralized, TDM trunks
- CUBE deployments with Unified CVP - centralized, outbound dialer

CVP deployments typically involve a VXML function and optionally a TDM to IP conversion function. CVP deployment recommendations sometimes provide for combining those two functions on the same ISR router. There are also CVP deployments that involve incoming SIP Trunks rather than TDM lines. These deployments may use CUBE routers and they may also host the VXML function. CVP also includes an optional component—Call Survivability—that allows branch routers to continue to provide a degraded level of service

to callers even if the WAN connection between the router and CVP is out of service. This component is implemented as a TCL-IVR application installed directly on each gateway and associated with a dial peer.

CVP deployments with CUBE media forking must manage up to four distinct activities:

- TDM to IP conversion
- Call Survivability
- Media forking
- VXML browsing

Some of these activities conflict with each other at the dial peer level, and certain steps must be taken in order to ensure that they interact well together. For example, you must not configure both media forking and a TCL or VXML application on the same dial peer. Each activity uses resources on the router, so they must all be taken into consideration for sizing. It is technically possible to configure one router to provide all four capabilities and in low call volume scenarios it is fine to do so. But as call volume rises, you must move either VXML browsing or media forking to a separate device. These two functions must not be co-located.

The function to isolate depends on your needs. VXML takes the bulk of router resources (especially if Automatic Speech Recognition is being used) and its sizing calculation is based on a different (usually smaller) quantity of calls than are the other activities. For the convenience and simplicity of sizing calculations, isolating VXML is a good choice.

However, if your intent is to capture only the agent part of the call in your recordings (see "Omitting the VRU segment from a recording") , then the configuration required to do so is far simpler if you perform media forking on a separate router. This has a further advantage in that co-locating TDM-to-IP, Call Survivability, and VXML browsing on a single router is the most common configuration for branch offices in a CVP deployment.

In multi-site ingress deployments, especially branch office deployments, you must use a combination of "Significant Digits" and "Send To Originator" functions in CVP's call routing configuration in order to prevent calls from inadvertently traversing a WAN link.

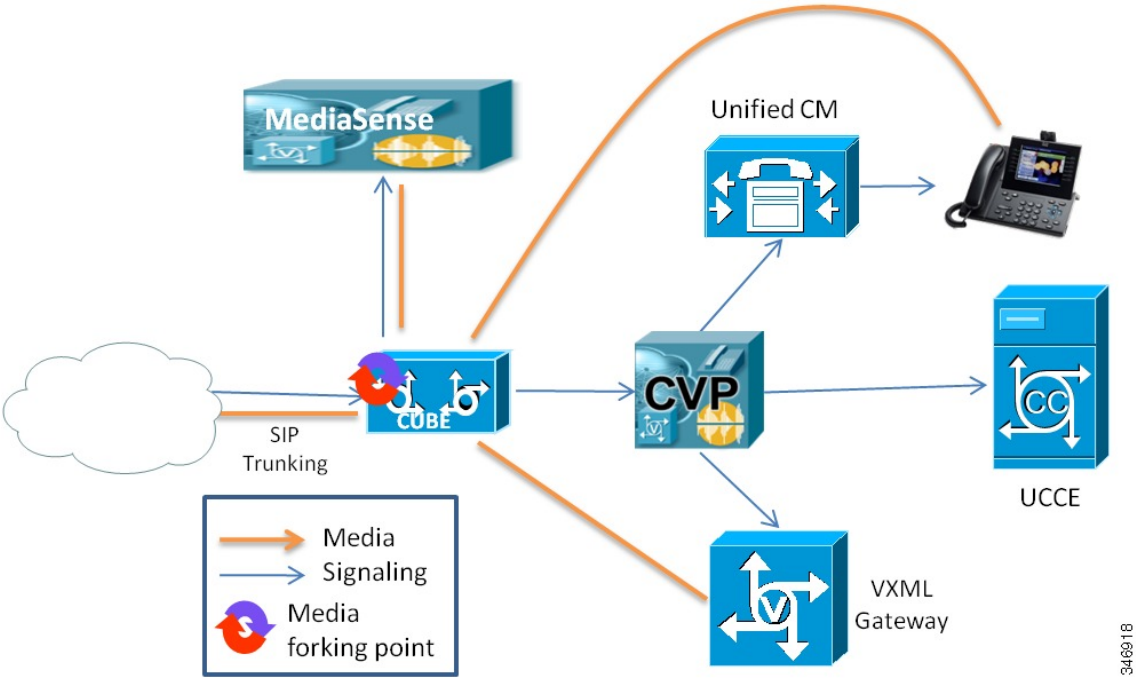
See the CVP documentation for more information about these techniques.

Note During normal processing of SIP messages, CVP inserts arbitrary data into the SIP content as a multi-part body. This format is currently not supported by MediaSense, nor is the content of interest to MediaSense. The recording dial peer in CUBE must be configured to prevent this content from being forwarded to MediaSense by adding the command "signaling forward none" to the recording dial peer.

If the same physical router is being used for both MediaSense and Unified CVP, it must be running a version of IOS which has been qualified by both products.

Except in the simplest of scenarios, contact the ISR sales team for capacity planning.

CUBE deployments with Unified CVP - SIP trunks, no survivability

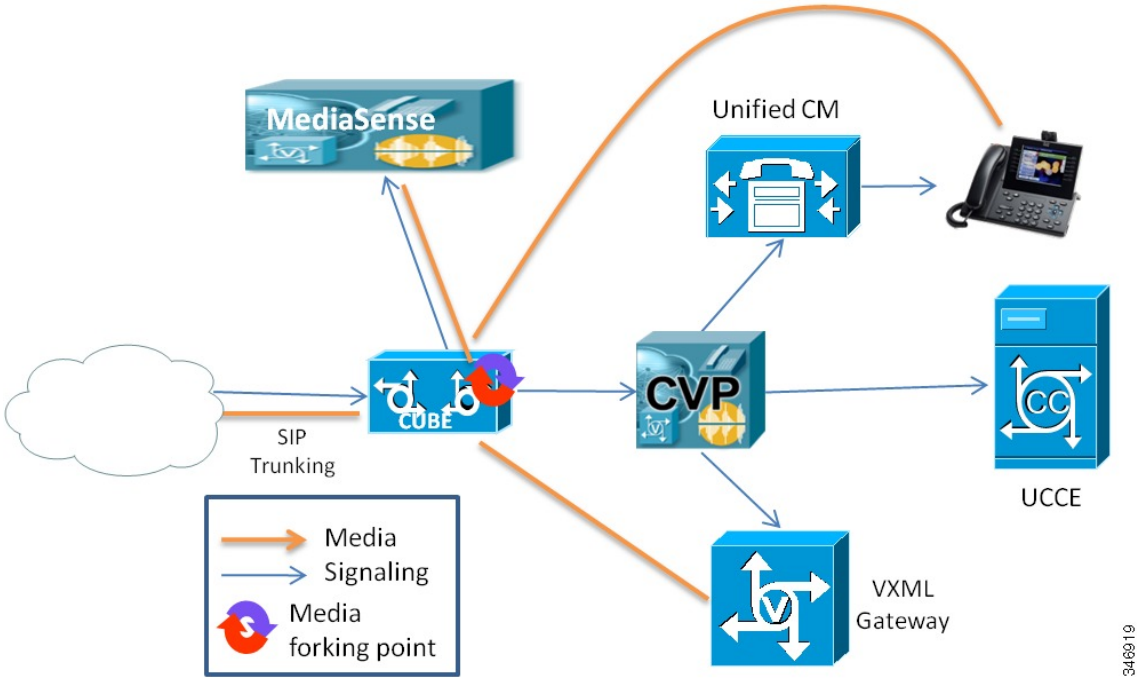


In this scenario, Unified CVP manages all call control operations including an initial delivery to a VXML gateway for music on hold or other treatment, a subsequent delivery to a Unified Contact Center Enterprise (Unified CCE) agent, and possible further network transfers to other agents and devices. All segments of the call are recorded.

When properly configured, Unified CVP affects these transfers by issuing SIP invitations to the destination device rather than to CUBE. This effectively re-routes the media without triggering a new dial peer match in CUBE.

As with most scenarios, media forking is configured on the outside dial peer.

CUBE deployments with Unified CVP - SIP trunks, with survivability



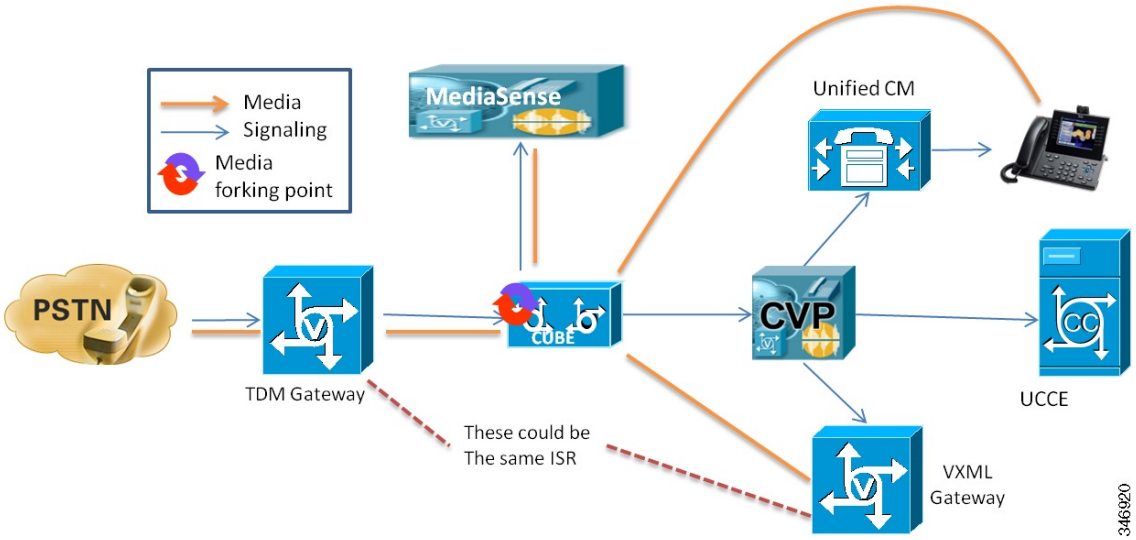
This scenario is identical to the preceding one except that the customer has elected to use the Unified CVP Survivability script to manage call failures and time of day routing. To use the Unified CVP Survivability script, place it on the outside dial peer in CUBE. IOS does not allow a script and media forking to occur on the same dial peer, however, so use the inside dial peer for media forking (as shown in the diagram). Configuring recording on the inside dial peer is risky because of the possibility that call manipulation may inadvertently trigger IOS to start a new dial peer matching operation. This would terminate the current recording session.

When properly configured, Unified CVP affects these transfers by issuing SIP invitations to the destination device rather than to CUBE. This prevents CUBE from triggering a new dial peer match.

Note If Survivability kicks in to handle a mid-call failure of any kind, any audio played by that script (such as a "technical difficulties" message) cannot be recorded by MediaSense. But if the script transfers the call to a local phone, that conversation can be recorded if the local phone's dial peer is configured for media forking.

For information about REFER transfers, see the section "Additional deployment options and considerations".

CUBE deployments with Unified CVP - TDM trunks

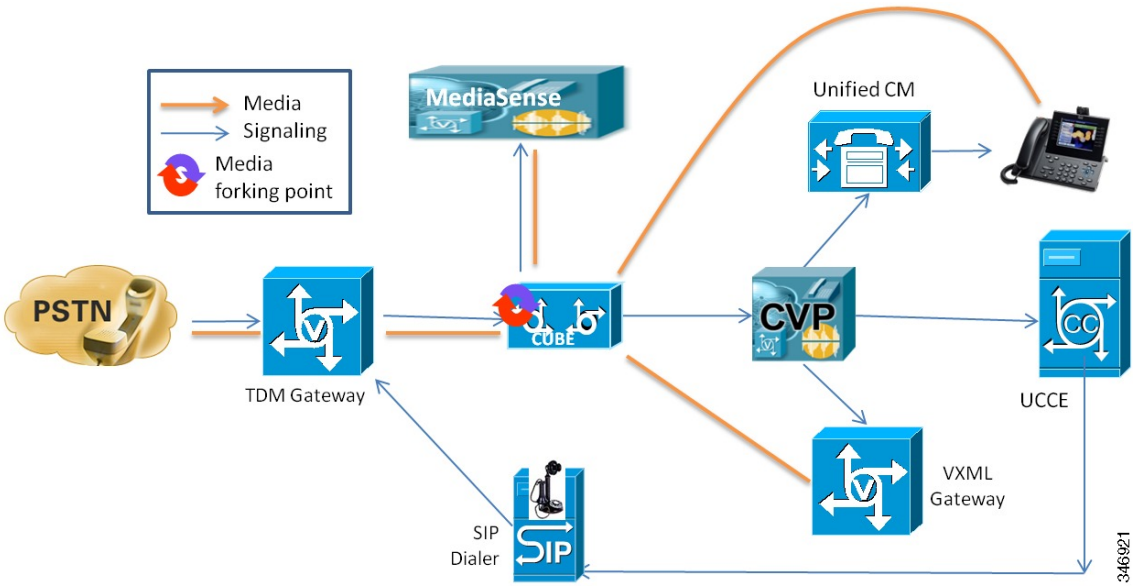


A TDM MediaSense CUBE deployment for CVP is just like a SIP trunk deployment, except that a logically separate TDM gateway is placed ahead of the CUBE. CUBE still does the media forking on the outside dial peer and CUBE still acts as the router that Unified CVP interacts with.

If Survivability is used, it is placed on the POTS dial peer in the TDM gateway; not in the CUBE. This keeps the media forking on the outside dial peer in CUBE.

If Unified CVP is issuing DTMF tones to the PSTN (as in "*8 Transfer Connect" transfers), configure either "dtmf-relay sip-kpml" or "dtmf-relay sip-notify" on both ends of the call connection between the TDM gateway and the CUBE.

CUBE deployments with Unified CVP - outbound dialer



Outbound campaigns using the Unified CCE SIP outbound dialer are configured to directly instruct the TDM gateway to call the target phone number. Once a party answers and the answering machine detection algorithm

determines that the answering party is a real person, the dialer instructs the TDM gateway to connect the call using CUBE to Unified CVP. From the perspective of CUBE and MediaSense, this appears the same as any another inbound call.

The outbound dialer is connected to the TDM gateway; not to the CUBE.

Additional deployment options and considerations

Redundant media forking using CUBE

Normally, one would apply the recording profile to the outside dial peer - the one which represents the side of the call which is external to the enterprise. It is also possible to configure media forking on both dial peers in a given call. This results in two independent recording sessions. The dial peers must be configured to deliver recordings to two separate and independent MediaSense clusters, implementing true recording redundancy. However, doing so severely impacts the performance of the CUBE. For sizing purposes, the CUBE call-carrying capacity should be assumed to be cut in half.

Percentage recording

Compliance recording, by definition, means that every call gets recorded. However some applications do not require that 100% of calls be recorded; in some cases spot-checking is sufficient.

Using CUBE, it is possible to record a pseudo-random sample of calls. This is accomplished by configuring multiple identical dial peers, assigning them equal preference values, but only configuring a subset of them for media forking. For example, one could record roughly one out of every three calls by configuring three identical inbound dial peers at preference level 5 and configuring media forking for only one of them.

Omitting the VRU segment from a recording

This applies to contact center recording where Unified CVP is used for call routing.

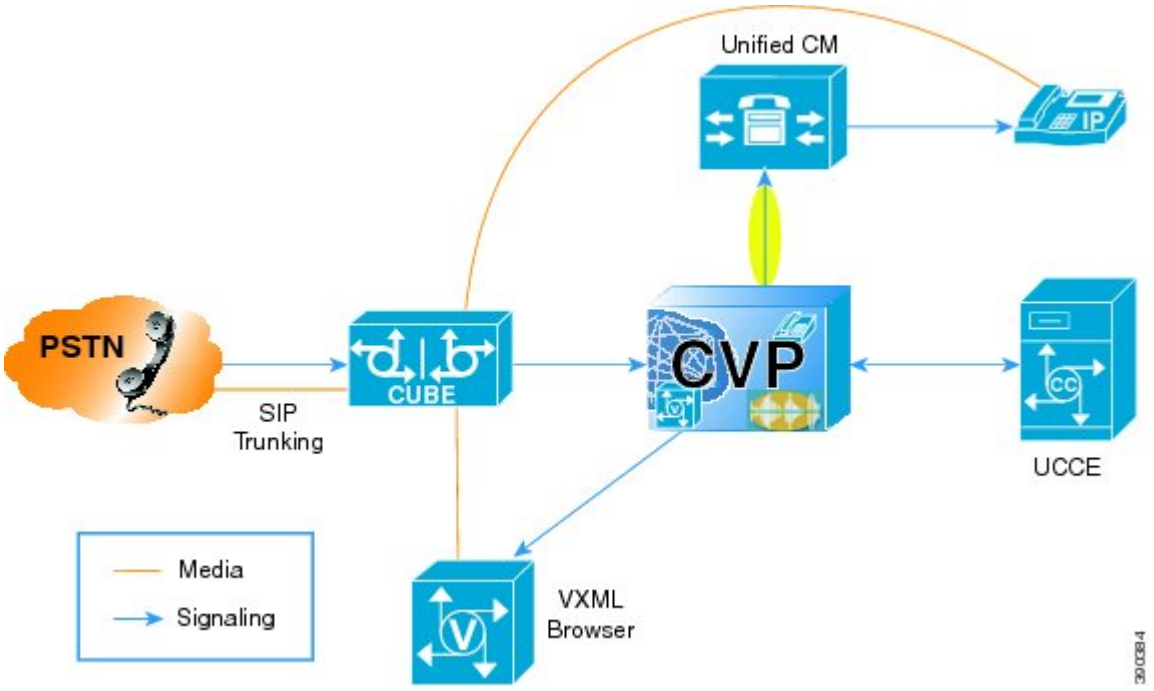
By forking media from CUBE, you can record the entirety of the caller's experience. This includes not only his or her conversation with one or more agents, but also any VRU or call queuing activity that may occur before the call is ever delivered to an agent. It can even be used to record the VRU activity if no agent is ever included in the call.

Some customers may want to omit the pre-agent VRU activity from the recording, particularly if it consists primarily of music on hold. One way to do this is by forking media from the agent's phone rather than from CUBE. But, if you need to fork media from CUBE for other reasons, you can accomplish this by causing Unified CVP to route the agent segment of the call back through the CUBE. You need to separate the ingress and media forking function from one another to do this, which means that you must either route the call through the ingress router a second time, or route it through a second router.

Both routing approaches require more hardware, but using a second router makes the configuration considerably easier. If your PSTN connection is TDM-based, you must route calls through the router a second time (or route them though a second router) anyway. Therefore, the remainder of this section assumes that the media forking router is separate from the ingress router, that the ingress router can be either a TDM gateway or an IP-only CUBE, and that the VXML function is running on the ingress router.

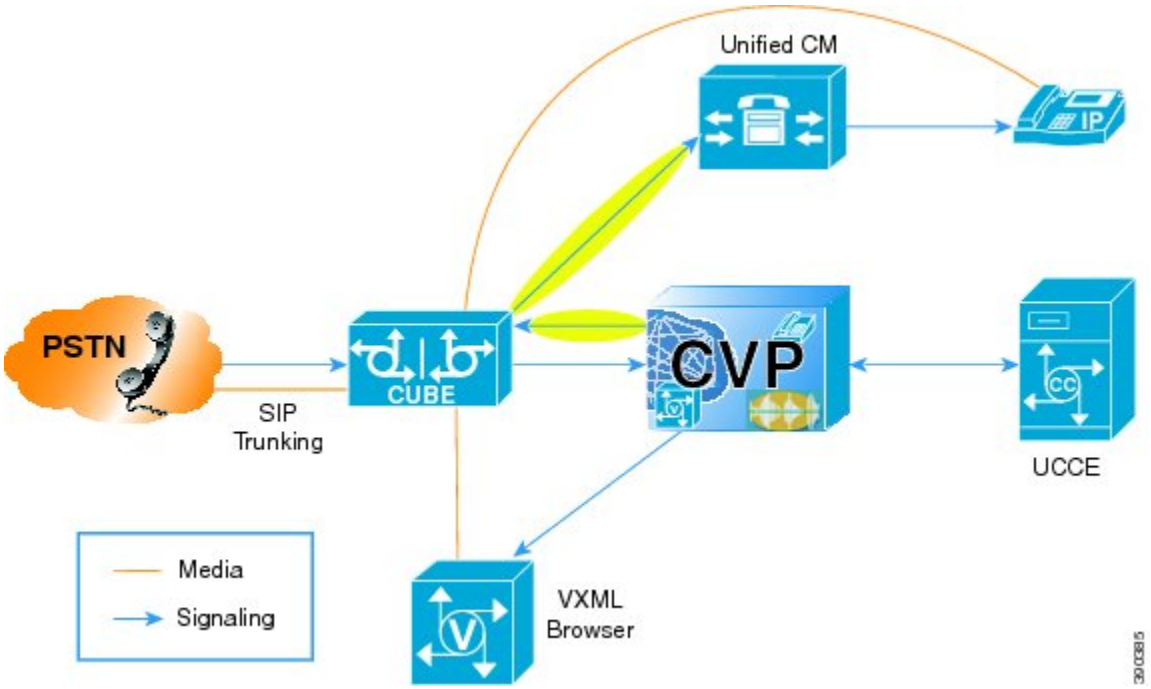
With a normal Unified CVP configuration, when an agent becomes available, Unified CVP sends a SIP invitation to the Unified Communications Manager that controls that agent's phone. Unified Communications Manager negotiates with the ingress router to connect the media stream from the router to the agent's phone. The ingress router itself never gets involved in routing that segment of the call because it never needs to figure out what IP address handles the selected agent's extension.

This arrangement is shown in the diagram below.

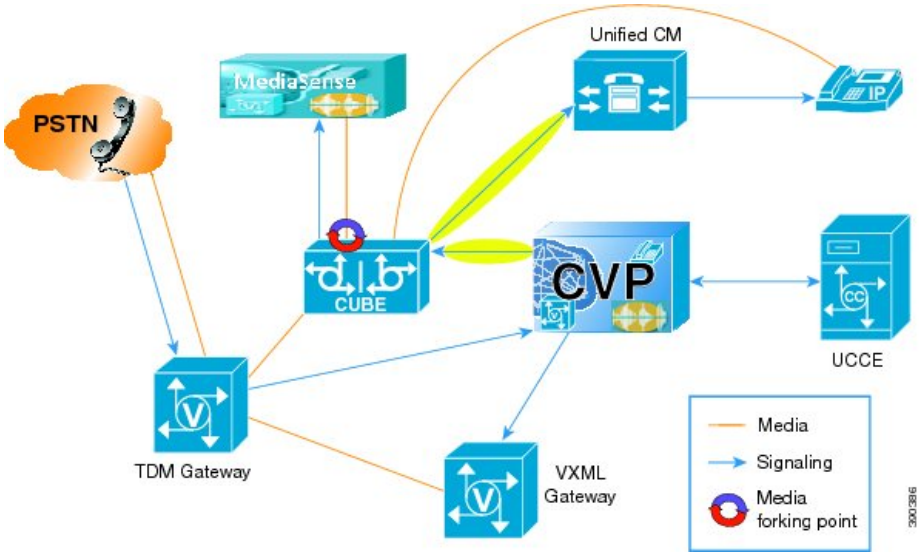


Unified CVP can also be configured so that the agent-segment invitation gets sent to the ingress router rather than to the Unified Communications Manager. The configuration can be done using Local Static Routes, an Outbound Proxy Server, or with Locally Resolved DNS SRV. One thing that will NOT work is checking the Enable Send Calls to Originator box in CVP's Dialed Number Pattern Configuration; that setting is only observed during the SendToVRU operation; not during the delivery to the agent. Once Unified CVP is so configured, you can define a dial peer in the ingress router that is specifically for routes to agent extensions—with Unified Communications Manager as the destination target.

This arrangement is shown in the following diagram.

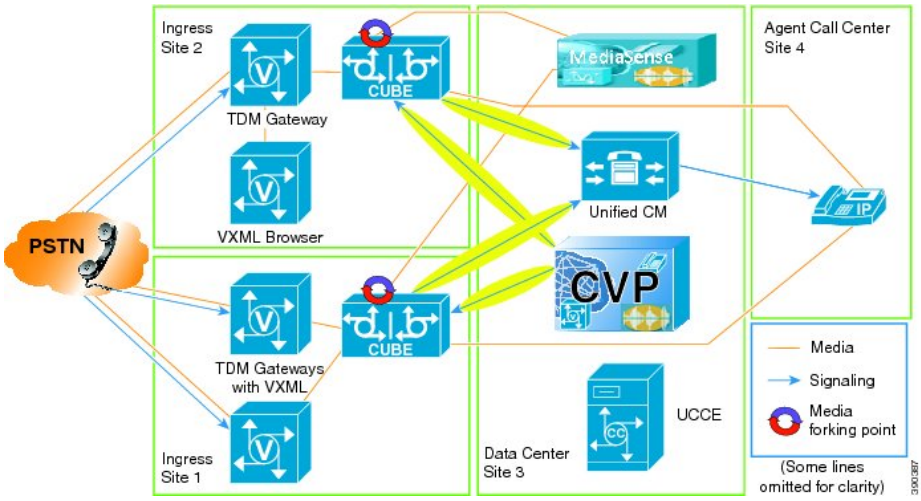


To add media forking, insert a second router - a CUBE - to do the media forking, as shown in the following diagram.



The situation becomes more complex when you have multiple ingress sites, but the goal is still achievable using a combination of CVP's "Send Call to Originator" and "Significant Digits" capabilities to avoid hairpinning calls across the WAN . Send Call to Originator allows CVP to ensure that any given call's own ingress router is where its VXML activity is performed. Significant Digits can be used to ensure that when the call is delivered to an agent, it passes through a CUBE that is in the same site as the call's own ingress router. Significant Digits can also be used to localize VXML activity to any VXML-capable router at the ingress router's site, rather than being limited to the ingress router itself. The following diagram shows the final arrangement in a multi-ingress site scenario. In one site, we show two ingress gateways and one CUBE for media forking. The two ingress gateways are identical; both are performing both TDM-to-IP conversion and VXML functions.

In the other site we show the same number of routers, but one router is used for TDM-to-IP conversion and a second router is dedicated to VXML activity.



Regardless of the configuration, bandwidth usage must always be considered. In the design in the diagram immediately above, media flows twice over the WAN: once to and from the agent's phone, and a second time from the media forking CUBE to the MediaSense cluster. If you co-locate MediaSense with the CUBE, there is no problem. But if your deployment calls for centralizing MediaSense in a shared data center, then you must consider this extra bandwidth usage. In order to avoid the extra WAN traffic, you could also move the media forking CUBE to the data center where MediaSense is located. This can only work if your Unified Communications Manager cluster and your agent's phones are all in the same WAN location. Otherwise, you will end up causing more WAN traffic rather than less, since you cannot force calls to pass through a CUBE which is co-located with the selected agent's phone. Media streams will frequently hairpin first through a CUBE that is located where the agent is not. This technique also has the potential to confuse Unified Communication Manager's Call Admission Control (CAC) algorithm.

REFER transfers

By default, CUBE will pass a REFER transfer from Unified CVP on to the next upstream user agent. Once that transfer succeeds, CUBE is no longer in the signaling or media path and therefore cannot further record the call. If your deployment environment permits it, you can configure CUBE to "consume" the REFER transfer rather than pass it on. This results in CUBE itself executing the transfer, taking Unified CVP out of the loop, but keeping CUBE itself in the signaling and media path and recording the call. You can accomplish this by adding "voice service voip; no supplementary-service sip refer" to your CUBE configuration.

Note If the inside dial peer is doing the media forking, then a REFER will always terminate the recording because it forces IOS to perform a new dial peer match operation.

Combining deployment models

The deployment models described in this document are not exclusive of each other. Any typical installation may have some inbound calls, some outbound calls, some that use Unified CVP, some that are not part of a contact center, some that use TDM trunks, some that use SIP trunks, some that fork media in CUBE, some that fork media at the phone, and so on. Generally speaking, the models here should be seen as describing the path that any one particular call may follow, while other calls may follow the paths which are covered in other

deployment models. In that sense, all of these models may be combined indiscriminately, as long as any single call remains within one single model.

Combining TDM to IP conversion with media forking

By definition, only a SIP to SIP call may fork media in a CUBE. However, there is no reason that one cannot insert T1/E1 cards into an ISRG2 running CUBE software. Calls that arrive on a TDM port can be recorded if they are routed through the device twice: once as a TDM to SIP call, and once as an SIP to SIP call. This can be accomplished by configuring the device's outbound dial peer of the TDM to SIP call to specify itself in its session target parameter. Using some digit manipulation or other means of qualifying the call, the second time the call arrives it matches a different (VOIP) dial peer and looks like a SIP to SIP call. On this second pass through the router, media forking can be enabled.

In this flow, the call gets handled by the router twice, and therefore counts as two calls from a capacity perspective. Put the other way around, *calls which follow this flow will effectively halve the stated capacity of the router*, thus requiring twice as much router capacity for the same number of calls. If you intended to use the full capacity of the router for calls, you need two routers. This presents a choice: when you double the number of routers, you can either have all routers do both jobs (TDM and forking), or have half the routers do each job. Either approach may be used if engineered correctly.

For more information about ISR configuration, see https://supportforums.cisco.com/docs/DOC-23115.

Combining VXML with CUBE media forking

In Unified CVP deployments without MediaSense, it is possible to run VXML voice browser functions on the same router as CUBE, but Cisco does not support sharing media forking and VXML activities on the same router. VXML, especially with automatic speech recognition, uses a lot of the router's resources. It also makes for a complicated sizing exercise, since for media forking you must consider the total number of concurrent calls being handled, whereas for VXML you only consider the number of concurrent calls that are expected to be in queue or in IVR scripts.

In multi-site ingress deployments, especially branch office deployments, this means that CVP must be configured to use "SigDigits" functionality rather than "SendToOriginator", in order to prevent calls that are in queue from inadvertently traversing a WAN link.

See the CVP documentation for more information about these techniques.

When to use CUBE and when to use a built-in bridge (BiB)

For call recording to work from a solution perspective, it is not enough for MediaSense to be able to capture the forked media. Third party applications must also be able to find those recordings by correlating identifiers that they know about with the identifiers that MediaSense knows about.

Calls recorded using a built-in bridge (BiB) are usually no problem, since the third party application usually has a JTAPI interface it can use to find the xRefCi values. But calls recorded using CUBE only receive the Cisco-GUID (which it exposes as "CCID" - Call Correlation ID), but JTAPI does not expose the Cisco-GUID at all. In Unified CCE environments using CVP, the third party application can only get the Cisco-GUID for inbound calls using the CTIOS event stream.

For large scale deployments:

- For Unified CCE environments:
    - Record all calls using BiB; or

- Record inbound calls using CUBE and record internal (consult) and directly dialed outbound calls using BiB.

- For Non-Unified CCE environments:
  - Record all calls using BiB.

For smaller scale deployments, particularly those where the customer will be using the built-in Cisco MediaSense Search & Play portal to locate and playback recordings:

- For Unified CCE environments:
  - Record inbound and outbound calls using CUBE and record internal (consult) calls using BiB.

- For Non-Unified CCE environments:
  - Record inbound and outbound calls using CUBE, and internal (consult) calls using BiB.

Exceptions and considerations:

- Unified CCE has a component (the SIP Dialer) for making automated outbound calls. Those calls end up looking like normal inbound calls as far as CUBE and MediaSense are concerned, so SIP Dialer calls can be treated as if they are normal Unified CCE inbound calls.
- Some partners or customers may be willing to use a less deterministic match than an explicit call identifier (such as the agent extension and the time frame). In these cases, the customer could use CUBE in all of the cases above—where BiB would otherwise be supported. (This is the basis for the relaxed recommendations for smaller scale deployments.)
- For internal (consult) calls or any calls between two Unified Communications Manager phones, BiB is preferred even if the customer is willing to use a less deterministic match than an explicit call identifier. It is possible to force these calls to hairpin through a CUBE, but the Unified Communications Manager configuration required to support this introduces restrictions on other Unified Communications Manager functions. Therefore, this type of configuration is not supported by Cisco.
- Unified CCE Mobile Agents raise additional considerations. BiB recording is not possible with these agents, but for inbound Unified CCE calls, they can be recorded with CUBE as described above. Consult calls to other agents can be recorded with BiB only if the other agent has a BiB-capable device (for example, he or she cannot be another Mobile Agent). Consult calls from one Mobile Agent to another Mobile Agent can only be recorded by forcing the call to hairpin through a CUBE. Directly dialed outbound calls from Mobile Agents can only be recorded using CUBE and can only be correlated using a non-deterministic approach (agent extension or dialed number, plus time frame).

Configuration requirements for other solution components

This section lists any configuration requirements that may affect how a particular deployment is designed or how components are ordered. For detailed configuration instructions, see the *Cisco MediaSense User Guide*.

Unified Communications Manager

Unified Communications Manager must be configured appropriately to direct recordings to the MediaSense recording servers. To do this, you must configure a recording profile as well as various SIP parameters. Phone

zones must be configured to avoid the use of iLBC or iSAC codecs and the Unified Communications Manager AXL service must be enabled on at least one of its servers (because MediaSense uses AXL to authenticate users.

Note SIP over UDP is not supported for MediaSense.

CUBE

Cisco Unified Border Element (CUBE) software with media forking runs only on Cisco ISRG2 routers. Different models have different scalability specifications, but it is always advisable to provision these routers with the maximum amount of memory available. The 3945E in particular requires a minimum of 2GB memory. Media forking is not supported on ASR routers.

Every MediaSense CUBE deployment requires an AXL connection to a Unified Communications Manager for authentication purposes, even if it will not be processing calls. The connection can be to a Unified Communications Manager that is already installed and in use for other purposes, or it can be one that is installed specifically for use with MediaSense. The administrator configures one or more Unified Communications Manager end users and imports them into MediaSense as MediaSense API users.

Streaming media players

Examples of off-the-shelf media players include:

- VLC version 2.0.8
- QuickTime
- RealPlayer

Each of these media players has its own advantages and disadvantages. VLC, for example, can only play one media track at a time. Quicktime is sometimes not able to handle the necessary authenticated RTSP redirect. Also, be aware that none of these media players are designed to handle silence. Playback of recordings that include silent segments may produce unpredictable behavior.

None of these players support AAC-LD, g.729 or g.722 codecs. A custom media player is required in order to play media that was recorded using those codecs. The built-in MediaSense media player, accessible through the Search and Play application, can play all of these audio codecs except AAC-LD.

Cisco does not produce, recommend, or support the use of these or any other third party media player. The only media player that Cisco supports is the one that is built in and provided by MediaSense.

SIP proxy servers

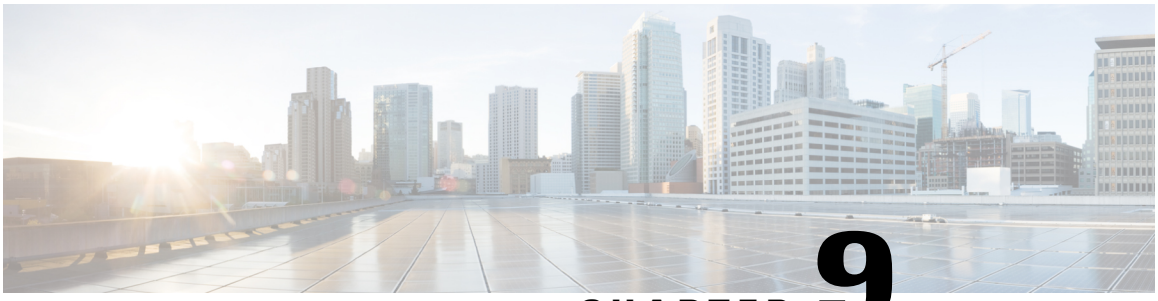
SIP proxy servers are currently not supported between MediaSense and Unified Communications Manager or CUBE.

Cisco Unified Session Manager Edition

In Cisco Unified Session Manager Edition (CUSME) deployments, MediaSense may only be placed at the "leaf" Unified Communications Manager cluster level. It is not currently supported at the centralized CUSME level. This means that each leaf cluster requires its own MediaSense cluster.

Contact Center Environments

- MediaSense does not explicitly interact with or support Unified Contact Center Enterprise (Unified CCE) or Unified Contact Center Express (Unified CCX). The recording functions that are available with the Agent and Supervisor Desktop clients on these products use different mechanisms for initiating and capturing recordings and require their own established recording solutions.
- For the Whisper Announcement feature, MediaSense does not record the whisper call between agent and supervisor (because the agent phone build-in-bridge normally doesn't include the supervisor-to-agent whisper in the forked media stream that it delivers to the recorder). On the other hand, if the supervisor phone is configured for forking, the whisper announcement is included in the supervisor phone recording.
- Equipment which monitors agent conversations by listening to a span port output and filtering on the agent phone MAC or IP address may not function properly when the phone is forking media for recording. This is because every RTP packet is emitted from the phone twice, and the listening device may not exclude those packets that are destined for the recording server from its capture. This results is the listener hearing an echo and needs to be taken into account for silent monitoring if the application calls for monitoring of conversations that are also being recorded.



CHAPTER 9

High availability

MediaSense implements a redundant, high availability architecture. Under normal operation, all deployed servers are always fully active. The following sections describe various aspects of this design.

Recording server redundancy - new recordings

A MediaSense cluster may contain up to five servers, each capable of recording up to a specific number of simultaneous calls. The method differs slightly for Unified Communications Manager and CUBE calls.

Conceptually, there are two phases involved. First, the call controller (Unified Communications Manager or CUBE) selects a MediaSense server (the pilot server) to send the initial invitation to. Second, the pilot server redirects the invitation to another server (the home server) to handle the call. Since any server may function as the pilot server for any call, the first phase is designed to prevent any single point of failure for the initial invitation.

The second phase allows MediaSense servers to balance the load among themselves without any active support from the call controller. The algorithm is aware of the state of all recording servers within the cluster and does not direct recordings to failed servers or servers with critically low disk space or other impacted conditions. It also ensures that the two media streams associated with a given call are recorded on the same server.

Unified Communications Manager is configured so that it sends invitations to each server in succession, in round-robin fashion. This ensures equal distribution of initial SIP invitation preference to all recording servers and avoids situations where one server receives the bulk of the invitations. As CUBE does not support a round-robin distribution, instead it is configured to always deliver invitations to one particular MediaSense server, with a second and perhaps a third server configured as lower preference alternatives. If possible, it is

best to target an expansion server rather than a primary or secondary server for the pilot role because expansion servers are typically doing less work at any given time.

If any recording server is down or its network is disconnected, it cannot respond to the call controller's SIP invitation. The usual SIP processing for both Unified Communications Manager and CUBE in this case is to deliver the invitation to the next server in the preference list. However, the call controller must wait for at least one timeout to expire before trying another server.

Since Unified Communications Manager and CUBE only involve recording servers after the primary media path has already been established, such operations can take much too long for the resulting recording to be useful. (Unified Communications Manager sets a time limit beyond which, if the recording hasn't begun, it will stop trying.)

The result is that if Unified Communications Manager selects a recording server that is not responding, the call in question will most likely not be recorded. CUBE does not have such a time limit; therefore such calls will end up being recorded, but a substantial initial segment of the call will be clipped.

To reduce the likelihood of lost recordings due to a recording server failure, MediaSense works with Unified Communications Manager and CUBE to support a facility known as "SIP Options Ping". This facility enables the call controller to periodically probe each recording server to make sure it is up and running without having to wait until a call is ready to be recorded. Once the call controller is aware that a given MediaSense server is not running, it skips that server in the round-robin or sequential list of recording servers. However, in single-node deployments, SIP Options Ping is not recommended. Not only is it not helpful, but it can in fact result in unnecessary failure recovery delays.

The *MediaSense User Guide* contains instructions for configuring the SIP Options Ping facility as well as other CUBE and Unified Communications Manager SIP parameters.

From a sizing perspective, be sure to provision enough recording ports so that if one server fails, you still have enough capacity to capture all the expected concurrent calls and that there is enough storage space for recording session retention.

Recording server redundancy - recordings in progress

If a recording server fails, all calls that are currently being captured on that server are changed from an ACTIVE state to an ERROR state, and the contents are discarded.

Note The detection of the failure of the call and the subsequent state change to error may not occur for some time (in the order of an hour or two).

There is currently no capability to continue or transfer in-progress recordings to an alternate server.

Recording server redundancy - saved recordings

After a recording is complete, MediaSense retains the recording on the same server that captured it. If that server goes out of service, none of its recordings are available for playback, conversion, or download (even though information about them can still be found in the metadata).

Metadata database redundancy

The primary and secondary servers (the 'database servers' in this section) each maintain a database for metadata and configuration data. They also each implement the MediaSense API and include the capability to publish events to subscribed clients. Once deployed, the two database servers are fully symmetric; the databases are fully replicated such that writing to either one causes the other to be updated as well. Clients address their HTTP API requests to either server and use the alternate server as a fallback in case of failure.

Database server failure

If either the primary or secondary server fails, the surviving server remains available for use. Once the failed server returns to service, the data replication mechanism automatically begins its catch-up operation without any user intervention.

Depending on the duration of the outage and the amount of churn that occurred, the catch-up operation could take some time. The actual duration depends on many factors; but in tests, it has taken close to an hour to transfer 150,000 recording sessions.

If the failure lasts for an extended period of time, the system will raise an alarm and disable replication completely, then reestablish it when the failed server recovers.

During the recovery period, some irregularities and inconsistencies between the two servers may occur. Do not rely on the recovering server for API operations until the catch-up operation is complete. You can determine the state of the recovering server using CLI commands.

Event redundancy

An event is generated by an individual database server when specific actions take place on the server. For example, when a recording server begins a recording, it initiates a session record in one of the database servers. Although the database update is replicated to its peer, only that one database server generates the event. This holds true for all types of events-- from recording session events to disk storage threshold events.

A client cannot know ahead of time which server will generate the events it is interested in. Each client must subscribe to both database servers in order to be sure it receives all events (the two subscriptions may designate the same target URI).

MediaSense also provides the capability for each database server to subscribe to events that are generated by the other database server and forward them together to subscribers (a flag is included in the event body that identifies these forwarded events). This capability is enabled in the MediaSense administration facility. If forwarding is enabled, a client need only subscribe to one database server; but doing so may sacrifice reliability. If the client's chosen database server goes down, the client must quickly subscribe to the alternate server in order to avoid any missed events. This risk should not be underestimated, especially considering that there is no reliable way for the client to detect such a loss without periodically issuing subscription verification requests.

When a client receives an event, there is an implicit guarantee that the database update associated with that event has already been committed to the database on the server which generated the event. Clients that need to execute API queries should check the event forwarding flag to ensure that they are querying the database server that generated the event.

Uploaded video playback redundancy

The load balancing and redundancy mechanism that is used to distribute incoming recording calls is very similar to the one used to distribute incoming video playback calls. The calls arrive at a pilot node and are immediately redirected to a home node. MediaSense can play an uploaded video as long as at least one of its nodes can play it.

Typically, all nodes in a cluster are able to handle a request and uploaded videos are distributed to all nodes. However, it is possible for a node to encounter some sort of error during the distribution or processing phases, or even for one node to simply be slower than the others in performing these duties. Therefore, incoming media playback calls get redirected to a node that is in service, has available capacity, and is ready to play the specific video selected by the appropriate incoming call handling rule.

Throttling of requests for video playback is also similar to throttling of requests for recording audio. Like an audio recording attempt, MediaSense treats incoming video playback requests at a higher priority than RTSP play and monitoring requests. Even though RTSP play and monitoring requests are subjected to a lower capacity throttling threshold than recording and video playback requests, it is possible for a given node to accept RTSP requests while recording and video playback requests are redirected to other nodes.

Unified Communications Manager failure while playing uploaded videos

If Unified Communications Manager fails while MediaSense is playing back uploaded videos, the media stream remains active but SIP signaling is no longer available. With no SIP signaling, there is no way for MediaSense to know when the endpoint device hangs up; therefore, the video plays until it comes to the end and then it terminates and releases all resources.

However, in the case of videos configured to playback repetitively (if configured as such in an incoming call handling rule), the playback may never terminate. For those cases, Unified Communications Manager sends MediaSense a session keep-alive message twice every 30 minutes by default. If MediaSense does not receive two of these timers in succession, it assumes that Unified Communications Manager has failed and terminates the playback and releases all resources.

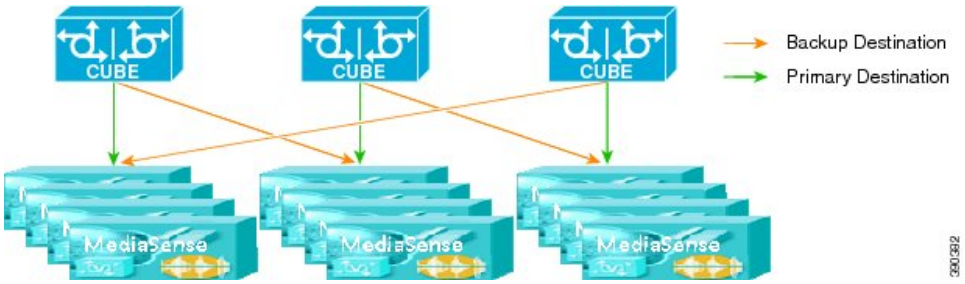
As a result, repetitive video play-backs can remain active for up to 30 minutes following a Unified Communications Manager failure. MediaSense considers those media streaming resources to be in use, or unavailable, for the purposes of determining whether new incoming calls and streaming requests can be accepted.

The 30 minute figure is a default Unified Communications Manager service parameter that can be modified.

MediaSense cluster redundancy

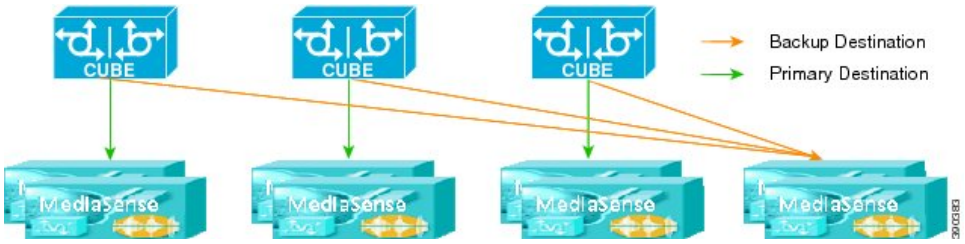
The individual nodes within a MediaSense cluster act in such a way so that if one node fails, the remaining nodes in the cluster automatically take up the slack—assuming they have available capacity. MediaSense clusters can also be configured to take over for one another in the event of an entire cluster failure. There are two general topologies that can be used, however in both cases, there is no load balancing across clusters. These are strictly hot standby arrangements in order to satisfy cluster failover requirements. They do not, for example, allow calls to be taken by one cluster if another cluster reaches its capacity.

Ring with spare capacity



In this topology, two or more clusters are arranged in a failover ring. Normally, all calls meant to be handled by Cluster A are handled by Cluster A, and ditto for Cluster B and Cluster C. However, the call controller (CUBE or Unified Communications Manager) is configured such that if it cannot send a call's recording to its usual target MediaSense cluster, it sends the recording instead to the next one in the ring. Calls for Cluster A would end up going to Cluster B, calls for Cluster B go to Cluster C, and calls for Cluster C would go to Cluster A. This requires that each cluster be provisioned with enough excess capacity to handle its own load plus the load on the preceding cluster. It is possible, but complicated, to configure failed-over calls to be distributed across all the remaining clusters, rather than only to the next cluster in the ring.

Spare cluster



In this topology, an entire extra cluster is provisioned and is not used except when one of the other clusters fails. Cluster D in this diagram is the spare one; Clusters A, B, and C are configured to fail over to Cluster D.

Configuration methodology

These two failover topologies use the same technique. They rely on SIP Options Ping (or rather the lack of it) to let the call controller know when an entire cluster is down. The technique works for both CUBE and Unified Communications Manager phone forking, but the configuration differs somewhat between the two.

For CUBE forking, each CUBE must be configured to fork recordings to two different nodes in the same cluster, followed by two different nodes in the designated failover cluster. Normally, all of the invitations first go to the targeted node in the first cluster and that node ensures that they get balanced evenly across all the nodes in the cluster. If the first targeted node goes down, it stops responding to SIP Options Pings. CUBE then stops sending invitations to it and sends them instead to the second targeted node in the first cluster. That node then ensures that the invitations get balanced across all the remaining nodes in the cluster.

If the entire cluster fails, then both of the first two nodes stop responding to SIP Options Pings. CUBE starts sending its invitations to the third targeted node, which is in the designated failover cluster.

Whenever any node comes back online, it starts respond to SIP Options Pings again and CUBE reverts to sending its invitations to that node, effectively restoring normal operation.

Note Configuring recording profiles in a round robin fashion (that is, successive invitations are delivered to successive configured nodes, with the first node in the sequence following the last) does not work for implementing cluster failover, but you can use Unified Communication Manager's top-down approach instead. You can configure the first two destinations as nodes in the first cluster, followed by two more nodes in the second cluster. Failover and recovery then will work just as they do in the CUBE scenario above.

Backup and restore

MediaSense does not provide its own built-in backup and restore capability, instead it permits the use of VM backup mechanisms for the system, metadata, and media disks. However, because of unpredictable performance impact, the VMWare virtual machine snapshot capability should not be employed with MediaSense except as part of the software upgrade process.

Note When using VM backups for MediaSense, it is important to know that VMs are only backed up on a node-by-node basis, but MediaSense functions in a cluster model.

Therefore, for example, when you backup an expansion node, you are not capturing any of the metadata for recordings on that node since the metadata is stored on the primary and secondary nodes. Similarly, if you backup the primary node, you are only capturing those recordings that physically reside on that primary node.

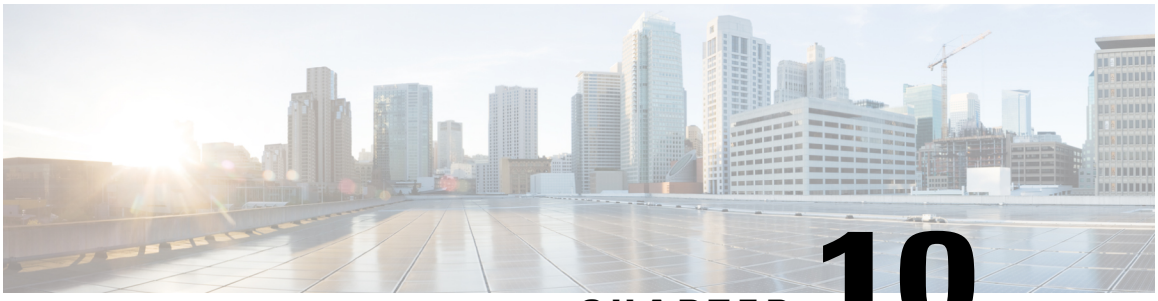
As in other normal restore scenarios, you can only recover information captured up until the last backup of that node (information captured after the last backup is lost). With MediaSense, recordings captured on that node since the last backup of that node are lost, but not their metadata. The metadata stored on the primary or secondary nodes (even if it is the primary or secondary node being restored) remains intact.

If you want to selectively preserve individual media files, convert them individually to .mp4 or .wav format and download them to another separate server for backup.

Network redundancy and NAT

Network redundancy capabilities (such as NIC Teaming) may be provided at the hardware level and be managed by the hypervisor. MediaSense itself plays no role in network redundancy and is completely unaware of it.

Network Address Translation (NAT) cannot be used between MediaSense and any peer device, including an API client machine. A given MediaSense node may only be known by one IP address. Various API responses and redirections include full URLs for accessing internal media resources. URLs that embed IP addresses that are not known to the clients using them will not work properly.



CHAPTER 10

Security

User administration and authentication

MediaSense supports three types of users: API users, application administrators, and platform administrators. There is only one application administrator and one platform administrator. Both of these users are configured during installation and the credentials for them are stored on MediaSense. Any number of API users can be configured after the installation process is complete.

For API users, MediaSense uses Unified Communications Manager's user administration. Any users configured as end users in Unified Communications Manager may be enabled as MediaSense API users. Once signed in to MediaSense, any such user can access all API functions. API clients sign in using a MediaSense API request, but MediaSense delegates the actual authentication of the user to Unified Communications Manager using the AXL service. API user passwords are maintained in Unified Communications Manager only and are not copied to MediaSense.

MediaSense does not currently support the notion of multiple roles and authorizations.

MediaSense APIs and Events

MediaSense API interactions are conducted entirely over secure HTTPS. All API requests must be issued within an authenticated session, denoted through a JSESSIONID header parameter. Authentication is accomplished through a special sign-in API request. However, SWS events are only delivered to clients using HTTP; HTTPS is not currently supported for eventing. By default, MediaSense uses self-signed certificates, but customers may install their own. When certificates are provided by clients, MediaSense always accepts them and does not verify their authenticity.

Internal intracluster communication

Components in a MediaSense cluster communicate with each other over unencrypted HTTP or Java Messaging Service (JMS) connections. The specifications for these interactions are not publicly documented, but they cannot be considered to be secure.

Media output URIs

A number of HTTP, HTTPS, and RTSP URIs may be associated with each recorded session. HTTPS URIs are secure by definition, but their security extends only to the transport mechanism. The URIs themselves can be transmitted insecurely by people or equipment.

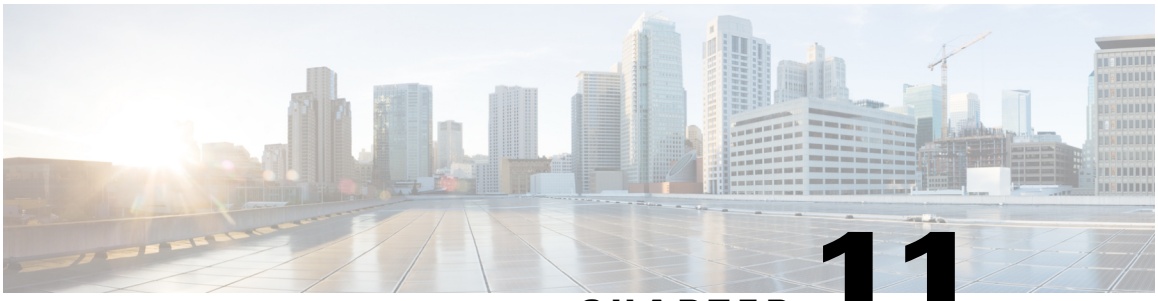
To prevent unauthorized users from making inappropriate use of these media output URIs, MediaSense requires that HTTP-BASIC authentication credentials be provided every time such a URI is used. In other words, a client must authenticate itself as a valid API user before it is given access to the recorded media. This authentication is usually very fast, but it may occasionally take up to 4 seconds to complete.

Uploaded media files

Administrator credentials are required to upload videos for ViQ, VoH and VoD purposes. The administration interface includes links that can be used to download previously uploaded MP4 files. Although administrator credentials are required to access the interface, the download links do not require credentials, and therefore cannot be considered as secure.

Media

Media encryption in transit, using Secure RTP (sRTP) or other means, is currently not supported. Media may however be stored on an encrypted SAN, as long as disk throughput requirements are met. Provisioning and configuring SAN encryption is outside the scope of MediaSense information.



CHAPTER 11

Reporting and call correlation

The information available in the MediaSense metadata database is limited to

- data provided in a CUBE or Unified Communication Manager SIP invitation
- tags that are inserted by client applications
- information generated within MediaSense itself.

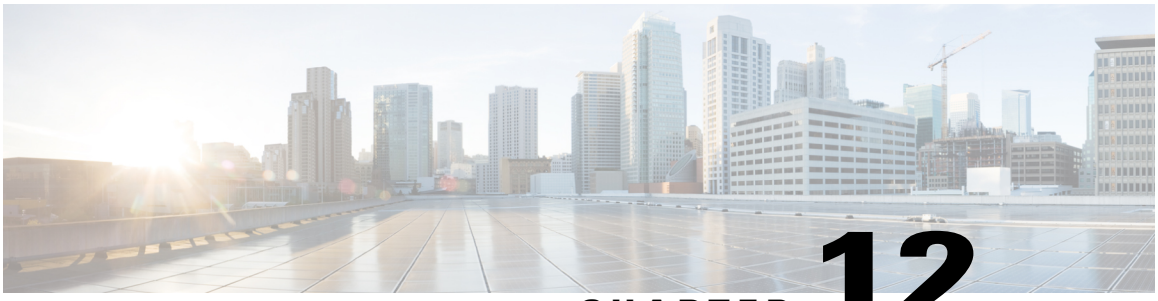
Real-time correlation with other components uses an identifier known as the Cisco-Guid. This identifier is usually created by an IOS device (such as CUBE or a gateway) or by the Customer Voice Portal (CVP) and is forwarded to other components that the call encounters. It is used to correlate calls across components either in real time or historically.

However, for Unified Communications Manager calls, MediaSense does not receive the Cisco-Guid.

Other identifiers are used to correlate recordings in MediaSense with historical call records in other solution components, but the only way to correlate recordings in real time is to have a TAPI or JTAPI connection with the Unified Communications Manager.

The device extension, which is available in both MediaSense and Unified CCE, can be used to associate data. But this can be problematic because lines are configured for recording in Unified Communications Manager, not devices or extensions-- and there is not necessarily a one-to-one correspondence between the line and the extension. If a given phone happens to have two lines or two extensions (or a given extension happens to be assigned to two phones), then some ambiguity can result.

For more information about call correlation techniques, see http://docwiki.cisco.com/wiki/FAQs_for_Cisco_MediaSense.



CHAPTER 12

Serviceability and administration

MediaSense offers a web-based user interface for administrative activities such as adding and configuring MediaSense servers, managing users, and checking and configuring storage management parameters.

The interface also provides access to support system serviceability functions that are required to service the product. MediaSense offers most of these functions through the Real Time Monitoring Tool (RTMT), which is similar to Unified Communications Manager and other Cisco voice products. RTMT is a thick client that can be downloaded onto any Microsoft Windows system (other operating systems are not supported) from the MediaSense serviceability web pages.

RTMT provides the following capabilities:

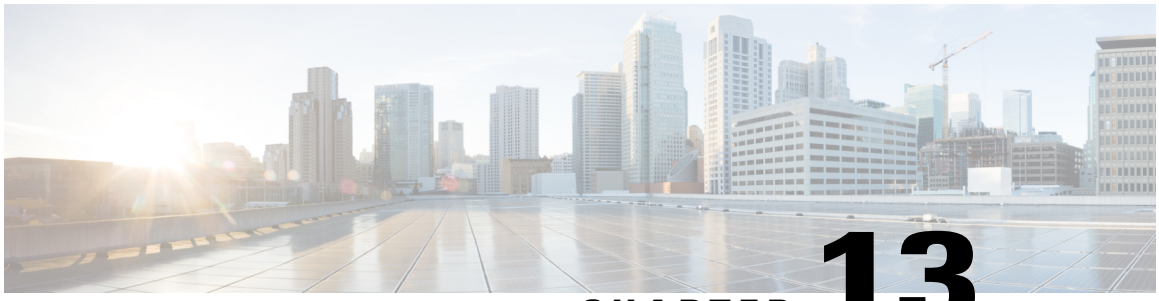
- Collecting log files of specific types and specific time periods from MediaSense servers. Remote log browsing is also available so that logs can be viewed without having to download them.
- Displaying alerts (including system conditions). System conditions are service impacting conditions such as the temporary or permanent outage of a server or critical subsystem, an overload condition, or loss of connectivity to a dependent service. Events which raise or clear system conditions may also be sent to a SYSLOG server or trigger proactive notifications to be sent by email.
- Displaying and graphing a large array of both system and application level counters, statistics, and performance measurements including the amount of space in use on the media partitions and the number of recordings in progress at any given time. RTMT also allows thresholds to be configured for these values that, when crossed, create an entry on the Alerts screen. As with system conditions, these alerts can also be sent to a SYSLOG server or trigger proactive notifications to be sent by email.

Separate from RTMT, MediaSense provides a specialized browser-based Serviceability user interface that provides administrators with the following capabilities:

- Starting, stopping, activating, and deactivating individual services.
- Selecting the level and type of information that gets written to log files.
- Requesting heap memory and thread dumps.
- Accessing other MediaSense servers in the cluster.
- Downloading RTMT for Windows.

MediaSense also supports a command line interface (CLI) for many additional service functions. Administrators of the Unified Communications Manager will already be familiar with most of these functions.

SNMP is not supported at this time.



CHAPTER 13

Design guidance

This section contains information intended to help plan for MediaSense deployment.

Proactive storage management

MediaSense offers both retention priority and recording priority storage retention modes. Under retention priority, clients are required to manage the space available for recordings because the system will not perform any automatic pruning. Under recording priority, the system will automatically prune old recordings but the pruning operation does not necessarily delete metadata or generated mp4 files that were created using the deprecated convertSession API. MediaSense can be configured to automatically clean up these elements or clients can take proactive responsibility for managing their disk space.

When using recording priority and you have not configured MediaSense to automatically delete pruned metadata, the client application must actively delete sessions that have been automatically pruned. Clients may either periodically issue an API request for pruned sessions or they may elect to receive session pruned events and explicitly delete those it no longer needs.

When using retention priority, there is no automatic pruning and the client is fully responsible for guaranteeing that enough disk space is available for new recordings.

Only if the system is configured for recording priority and automatic deletion of pruned recordings is turned on can the client avoid taking part in storage management.

These session management activities are invoked using the MediaSense API. (For more information, see the *MediaSense Developer Guide*.) If pruning activities are going to be performed regularly, schedule them for low usage periods in order to minimize impact on normal operations.

Media storage space provisioning

At the application level, MediaSense contains two kinds of media storage--each in its own directory location. Recording storage is located in a partition known as **/recordedMedia** and uploaded videos are located in a partition known as **/uploadedMedia**. These are logical partitions that are each made up of 1 to 16 virtual disks at the VM level. The virtual disks are mapped (using VMWare host configuration) to physical disks on the server or on a SAN. The physical disks are configured and managed using a RAID controller.

The physical disks must meet certain speed and throughput requirements that are described in the "Storage" section in the Compatibility matrix that follows. ESXi "thin provisioning" is not supported on any disk.

SIP configuration

The following guidance applies to CUBE deployments

- In CUBE deployments, use 'SIP early offer' on dial peers that go to MediaSense. This is the default setting. Only Unified Communications Manager implements 'delayed offer' with no option.
- Use 'SIP over TCP' on dial peers or trunks that go to MediaSense.
- Configure 'SIP options ping support' for dial peers or trunks that go to MediaSense (except in single-node deployments). This feature greatly improves failover support for multi-server MediaSense deployments as well as for MediaSense cluster failover.

Codec configuration for phones

For Unified Communications Manager recording, some of the newer Cisco IP phones support iLBC or iSAC codecs and Unified Communications Manager may negotiate for them. However, since MediaSense does not yet accept these codecs, they must be disabled for recording enabled devices in the Unified Communications Manager service parameter settings.

Network provisioning

CUBE interfaces that carry RTP media *must* be configured to

- a fixed 1 gigabit speed, or higher
- be fully duplexed
- not rely on auto-negotiation

Sometimes auto-negotiation fails when 100 megabit speeds are available. Even when 100 megabit speeds are properly negotiated, they are not fast enough to handle a heavy call load.

Recording servers like MediaSense receive a lot of network traffic but generate relatively little of their own traffic. The asymmetric nature of such traffic can lead to the expiration of MAC address forwarding table entries on network switches, which may result in the network being flooded. Network administrators must take this possibility into consideration when configuring network switching equipment.

Using scalable queries

MediaSense offers an API for searching the metadata in a very flexible manner. While many queries will execute with little or no impact on the normal operation of the MediaSense servers, it is possible to formulate queries that have a significant impact. MediaSense limits the number of simultaneous queries it will process but does not consider the relative cost of each individual query. Customers who use the query APIs must read and adhere to the guidelines for writing scalable queries, which can be found in the *MediaSense Developer Guide*.

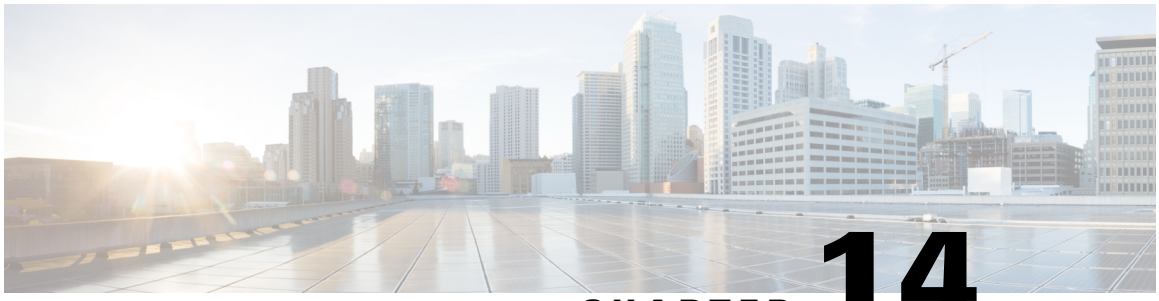
Distribute HTTP download requests over time

Some customers will use the HTTP Download facility to create copies of all recordings, using MediaSense more as a temporary location for these files than as a long term archive. Customers may also batch these requests and issue them once a day or on some other periodic basis. It is better, from a resource usage perspective, to distribute these requests more evenly over time. For example, use the session ENDED event to trigger a download as soon as a call recording terminates.

Alarm monitoring

Various situations that require administrator attention cause alarms to be raised (system conditions). These conditions are observed in the system logs as well as in RTMT's alarms page. Using RTMT, you can configure these alarms to be sent to a SYSLOG server and to send email messages to a designated email address. MediaSense does not currently support SNMP alarms.

At least one of these methods must be used to actively monitor the state of the MediaSense servers.



CHAPTER 14

Compatibility matrix

This section describes (in general terms) versions, model numbers, and specifications of components that MediaSense is compatible with.

For the up-to-date specifics, see the following wiki locations:

- http://docwiki.cisco.com/wiki/Virtualization_for_Cisco_MediaSense
- http://docwiki.cisco.com/wiki/Cisco_MediaSense_on_UCSE
- http://docwiki.cisco.com/wiki/Cisco_MediaSense_Compatibility_Matrix

Server platforms

MediaSense supports specification-based virtualization. Under this feature, Cisco extensively tests a number of specific hardware configurations (known as the tested reference configurations (TRC)), and then derives a set of specifications by which a partner or customer can select equivalent hardware models either from Cisco or from other vendors.

There are differences in the level of support that Cisco TAC provides for different hardware solutions. For more information, seehttp://docwiki.cisco.com/wiki/UC_Virtualization_Supported_Hardware.

A detailed list of TRC models and supported server specifications can be found at http://docwiki.cisco.com/wiki/Virtualization_for_Cisco_MediaSense . Other than the TRC models that are manufactured by Cisco, only Hewlett Packard (HP) and IBM servers are supported (subject to the stated minimum performance specifications).

Server configurations are divided into those that have direct attached disks (DAS) and those that don't. For diskless servers, you must provision fiberchannel SAN. For DAS servers, fiberchannel SAN is optional. It is important to ensure that the selected server can support sufficient disk space to house the amount of media storage required and that it meets the minimum disk configuration and performance specifications cited on the virtualization wiki.

When ordering C-series servers, be sure to include either battery backup or the Super Cap option for the write cache.

Hypervisor

A VMWare hypervisor is required. MediaSense is not designed to run on bare metal hardware.

For a list of supported hypervisors, see http://docwiki.cisco.com/wiki/Cisco_MediaSense_Compatibility_Matrix.

Storage

MediaSense uses storage for two distinct purposes. One set of disks holds the operating software and databases, and the other set is used for media storage. The two kinds of storage have very different performance and capacity requirements. Thin provisioning is not supported for any MediaSense disks.

Recorded Media Storage. Up to 60 terabytes is supported per cluster, divided into 12TB in each of five servers. This is the theoretical maximum, which could only be attained if you are using SAN storage. If you are using Directly Attached Disks (DAS), then you are limited to the physical space available in the server.

Uploaded Media Storage. Uploaded media requires much less storage, but can also support up to 60 terabytes, divided into 12TB in each of five servers.

If you are using Directly Attached Disks (DAS), then the first two disks (for operating software and database) must be configured as RAID 10.

If you are using SAN, note that only fiber-channel attached SAN is supported, and the SAN must be selected according to Cisco's specifications for supported SAN products (see "Cisco Unified Communications on the Cisco Unified Computing System" at http://www.cisco.com/go/swonly). Also, SAN storage must be engineered to meet or exceed the disk performance specifications for each MediaSense virtual machine. These specifications are per node. If the nodes are sharing the same SAN, then the SAN must be engineered to support these specifications, times the number of nodes. For security purposes, it is permissible to use an encrypted SAN for media storage as long as the specifications at the link below can still be met.

For information about current disk performance specifications for MediaSense, see http://docwiki.cisco.com/wiki/Virtualization_for_Cisco_MediaSense.

UCS-E router blade modules come with fixed disk hardware and MediaSense scalability limits for each type of module are designed with their actual performance characteristics in mind. You do not need to engineer their disk arrays to meet the specifications above. However, all of the drives should be manually configured as RAID-1.

Also, for these modules, the required downloadable .OVA template automatically carves the disks into two 80GB drives and one 210GB drive, formatted. For those modules that have additional disk space available, you may configure the additional space for either uploaded media or recorded media as best suits your application.

Other solution component versions

MediaSense depends only on the versions of Unified CM and IOS. There is no particular dependency on Unified CVP or Unified CCE.

However, for deployments that include both MediaSense and Unified CVP where the two products will be sharing the same router, the IOS version running on that router must be one that is compatible with both products. It is important to verify this in the compatibility matrix for each product at deployment time.

For current compatibility matrix information about MediaSense, see http://docwiki.cisco.com/wiki/Cisco_MediaSense_Compatibility_Matrix.

Phones

- For CUBE-based forking, all Cisco phones are supported; but for video calls only the audio portion is recorded.
- For endpoint-based forking (also known as Built-in-Bridge, or BiB forking), all Cisco phones that support BiB technology are supported, but you must ensure there is enough bandwidth available. BiB forking can result in up to 5 media streams :
  - two audio streams involved in the conversation (in and out of the user's phone).
  - two audio streams sent from the phone to the recorder (copies of the in and out streams).
  - one audio stream if silent monitoring.

Note No Cisco phones are currently capable of forking video.

- For direct recording, all Cisco phones are supported for both audio and video media.
- For outbound streaming of uploaded videos, any Cisco phone that can handle the audio codecs shown in the table below is supported, as long as it can also handle the video resolution of the uploaded video (the same is true for recorded video greetings in the Unity Connection integration). Most Cisco endpoints can automatically scale whatever resolution they receive, but some (such as the Cisco 9971) cannot down-scale.

The following table is a partial list of supported devices and codecs.

Endpoint category	Endpoint forking	CUBE forking	Direct recording	Outbound streaming for ViQ and VoH	Unity Connection video greetings	Models tested and verified
Audio hard phones	Audio (g.729, g.711µLaw and aLaw, g.722)	Audio (g.729, g.711µLaw and aLaw, g.722)	Audio (g.729, g711µLaw and aLaw, g.722)	Audio (g.729, g.711µLaw, g.722)	Not applicable.	All Cisco phones that support BiB. An up to date list may be found under "Unified Communications Manager Silent Monitoring Recording Supported Device Matrix" at http://developer.cisco.com/web/sip/wikidocs.
Cisco IP Communicator (CIPC)	Audio (g.729, g.711µLaw and aLaw, g.722)	Audio (g.729, g.711µLaw and aLaw, g.722) Audio streams from video calls (g.729, g.711µLaw and aLaw, g.722)	Audio (g.729, g711µLaw and aLaw, g.722) Video	Audio (g.729, g.711µLaw, g.722) Video	Not applicable.	Cisco IP Communicator (CIPC) v7.0(1) or later.
Cisco Jabber	Audio (g.711µLaw and aLaw)	Audio (g.711µLaw)	Audio (g711µLaw) Video	Audio (g.711µLaw) Video	Audio (g.711µLaw) Video (at maximum 640x480 resolution)	Cisco Jabber for Windows, Mac and iPad.

Endpoint category	Endpoint forking	CUBE forking	Direct recording	Outbound streaming for ViQ and VoH	Unity Connection video greetings	Models tested and verified
Video-capable phones	Audio (g.729, g.711µLaw and aLaw, g.722)	Audio (g.729, g.711µLaw and aLaw, g.722) Audio streams from video calls (g.729, g.711µLaw and aLaw, g.722)	Audio (g.729, g7l1µLaw and aLaw, g.722) Video	Audio (g.729, g.711µLaw, g.722) Video	Audio (g.711µLaw) Video (at maximum 640x480 resolution)	9971, 9951 and 7985. Video greeting only works with Cisco 9971 (or similar) phones using g.711 (uLaw or aLaw) and with h.264. See http://www.cisco.com/en/US/docs/voice_ip_comm/connection/10x/design/guide/ 10xcucdg070.html for a detailed list of supported phones.
Telepresence endpoints	Not supported	Audio (g.729, g.711µLaw and aLaw, g.722, AAC-LD) Audio streams from video calls (g.729, g.711µLaw and aLaw, g.722, AAC-LD)	Audio (g.729, g7l1µLaw and aLaw, g.722, AAC-LD) Video	Audio (g.729, g.711µLaw, g.722, AAC-LD) Video	Audio (g.711µLaw) Video (at maximum 640x480 resolution)	EX-60, EX-90, and SX-20. For recording, EX-60 must be configured for g.711uLaw/aLaw or g.722 due to CSCul00473. Other devices can support AAC-LD as well. AAC-LD media forking requires CUBE IOS 15.3(3)M1 or later. For outbound streaming, g.711aLaw is **not** supported, but AAC-LD is. CTS series is not supported for any purpose.

Web browsers

Web browsers are used for accessing the Serviceability and Administration functions on MediaSense servers. The following browsers are supported:

Browser	Version	Operating system
Internet Explorer	9	Windows 7 only
Firefox	24	Windows 7 and MacOS

When running the Search and Play application through one of the above browsers, a minimum version of the Java JDK or Java JRE must be installed, depending on the underlying operating system.

Underlying OS	Minimum Java version
Mac OSX	JDK or JRE 7 update 25, 64-bit
Microsoft Windows	JDK or JRE 7 update 25, 32-bit

MediaSense upgrades

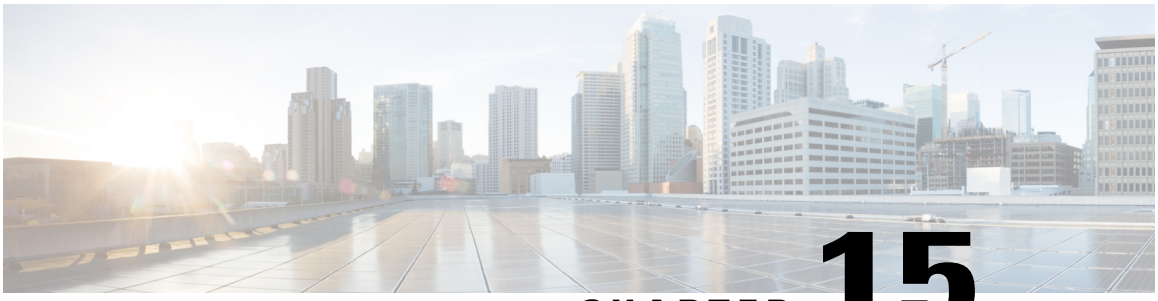
MediaSense can only be upgraded from one immediately previous release to the next. If you are upgrading from an earlier release, you will need to upgrade through each intervening version first. Upgrades from releases prior to 8.5(4) are not supported.

Each successive release contains minor changes to the MediaSense API, that are always upward compatible—but with one exception. The exception is between release 8.5(4) and 9.0(1), in which security enhancements were introduced. Those enhancements require that client software be modified in order to provide HTTP-BASIC credentials and to handle a 302 redirect. This applies to all RTSP streaming and HTTP download requests.

A new VMWare VM template was provided in release 9.1(1) that provisions 16 GB of memory rather than the 8 GB that was called for in release 9.0(1) and earlier. For any server being upgraded to 9.1(1), the VM configuration must be manually adjusted to reserve this increased amount of memory.

A new feature was added in release 9.1(1) that permits recorded media storage to be increased in size after installation. However, this feature is not available in systems upgraded from prior releases; it only functions in systems that have been fresh-installed with release 9.1(1) or later. The new uploaded media partition introduced in release 9.1(1) is automatically created during upgrade and does support the capability to be increased in size after installation.

If you upgrade a MediaSense cluster from 9.0(1) to 9.1(1) or later and then wish to add nodes to your cluster, be aware that though the new nodes will be installed with expandable recorded media storage, Cisco does not support that flexibility. Provision approximately the same amount of recording space on each new node as is available on each upgraded node. Although storage space disparity across nodes in the cluster does not present a problem for MediaSense, it could result in pruning ahead of the configured retention period on smaller nodes. Administrators may find this behavior unpredictable.



CHAPTER 15

Scalability and sizing

This section discusses MediaSense scalability and sizing.

Performance

The supported capacity for MediaSense is a function of the hardware profile that the system selects at startup time. The hardware profile depends on which VM template the node is deployed on, and the VM template depends partially on what type of hardware you are deploying. (See "Virtual machine configuration" for a full description of each template.) The "Hardware profiles" section below shows the actual capacity when using each type of VM template.

For example, for each "7 vCPU" template node (the standard for large production deployments) the MediaSense server supports up to 400 media streams simultaneously (200 calls) at a sustained busy hour call arrival rate of two calls per second on up to 12 terabytes of disk space. The 400 represents all streams used for recording, live monitoring, playback, .mp4 or .wav conversion, and HTTP download; all of which may occur in any combination. Conversion and download are not strictly speaking streaming activities, but they do use system resources in a similar way and are considered to have equal weight. Playback of a video track takes 9 times more resources than playback of an audio track. As a result, each uploaded video playback (one video track + one audio track) has the weight of 10 audio tracks, leading to a maximum capacity of 40 simultaneous video playbacks per node.

In determining how many streams are in use at any given time, you need to predict the number of onsets for each activity per unit time as well as their durations. Recording, live monitoring, and playback have a duration that is equal to the length of the recording. Video playbacks, if configured to play once only, have a duration equal to the length of the video. Video playbacks for hold purposes must be estimated to last as long as each video caller typically remains on hold. The .mp4 conversions, .wav conversions, and HTTP download durations are estimated at about 5 seconds per minute of recording.

To determine the number of servers required, evaluate

- the number simultaneous audio streams needed plus 10 times the number of videos being played, divided by the number of audio-weight media streams supported by each node
- the number of busy hour call arrivals divided by the maximum call arrival rate for each node
- the space required for retained recording sessions divided by the maximum media storage for each node.

The number of servers required is equal to the largest of the above three evaluations (rounded up).

Video playback for VoH, ViQ, and Video Messaging is further limited on 2\- and 4-vCPU virtual hardware and depends on the type of physical hardware being used. See the Hardware profiles section for details.

Another factor that significantly impacts performance is the number of MediaSense API requests in progress. This is limited to 15 at a time for 7-vCPU systems, with the capability to queue up to 10 more (the figures are reduced for smaller systems). These numbers are per node, but they can be doubled for MediaSense clusters that contain both a primary and a secondary node. For more information, see "System resiliency and overload throttling".

The media output and conversion operations (monitoring, playback, convert to MP4 or WAV, and HTTP download) are entirely under client control. The client enforces its own limits in these areas. The remaining operations (call recording and uploaded media file playback) are not under client control. The deployment can be sized so that the overall recording and video playback load will not exceed a desired maximum number cluster-wide (leaving room for an enforceable number of monitoring, playback, and HTTP download operations). The recording and video playback load is balanced across all servers. (Perfect balance will not always be achieved, but each server has enough room to accommodate most disparities.)

Hardware profiles

When MediaSense nodes are installed, they adjust their capacity expectations according to the hardware resources they discover from the underlying virtual machine. When the server is installed using one of the Cisco-provided OVA templates, the correct amount of CPU and memory are automatically provisioned and a matching hardware profile will be selected. The hardware profile determines:

- the number of audio-equivalent calls supported,
- the number of concurrent API requests supported,
- the maximum call arrival rate supported,
- the maximum number of nodes supported in the cluster,
- the maximum amount of media storage available,
- the cap on number of video playbacks supported, and
- a number of other internal parameters

as a function of the number of vCPUs, CPU speed, and amount of memory provisioned.

If an incorrect OVA template is used, or if the virtual machine's configuration is changed after the OVA template is applied such that the virtual machine does not exactly match one of the existing hardware profiles, the server is considered to be unsupported and the capacities in the "Unsupported" category are used.

For more information, see the Hardware Profile table at http://docwiki.cisco.com/wiki/Virtualization_for_Cisco_MediaSense.

Maximum session duration

MediaSense can record calls that are up to eight hours in duration. Beyond that duration, some sessions may end up being closed with an error status and HTTP download and .mp4 or .wav conversion functions may not succeed.

Storage

The amount of storage space required depends on a number of factors, such as the mix of codecs in use, the number of calls, the call arrival rate, duration, and duty cycle; and the retention period desired. Since most of these parameters are very difficult to estimate, the focus is on only the number of recording session hours and the retention period so that the amount of space required to retain h hours of recordings for d days can be accurately calculated.

Here is the formula:

Write Rate (W) = B * P * U, in hours of storage per hour of elapsed time

where

- B is the codec bit rate (in MB/hour for two streams)
- P is the number of phones
- U is the average usage ratio of each phone (in hours per day)

Retention (R) in hours = S * 1024 / W

where S is the total storage available across all servers (in GB).

Calls using the g.711 or g.722 codecs require about 1MB per minute of dual-stream recording. The space requirements for calls using the g.729 codec (that uses a variable rate compression) are not predictable, but generally speaking require about one eighth the space needed by g.711 calls (about 128 kilobytes per minute of dual-stream recording). H.264 video is even less predictable.

Therefore, assuming the use of g.711 or g.722 codecs, we have a rate of 1MB per minute or 60MB per hour. With maximum direct-attached storage of 4TB of disk space, we can store about 70,000 hours of dual-stream recordings. With 100 phones that are actively used 80% of the time (24 hours a day), then recording occurs at a rate of 80 hours per hour of elapsed time. This uses 70,000 hours in 875 hours of elapsed time (or a little over 36 days), after which the oldest calls need to be pruned. Therefore, your retention period is 36 days.

If all of the same parameters applied but phones are only active 12 hours per day, the retention period would then be 72 days.

The above examples are based on having 4TB of storage space available. If you deploy five MediaSense servers, each with its maximum SAN storage allocation, then there are 60TB of storage space available.

The number of files converted into .mp4 format using the deprecated convertSession API is another factor to consider when calculating storage space. If you expect to be converting a significant number of recorded sessions to .mp4 and leaving them on the server, then you must increase the Write Rate (W) to account for it.

In lab trials, files in .mp4 format averaged about 18 MB/hour for dual-channel audio, and about 180 MB/hour for audio+video. (Note that the .mp4 files use AAC variable rate encoding, so the actual space used may vary considerably.) If you convert and retain an average of 50% of your recorded sessions for example, then you must increase the Write Rate by 50% times the .mp4 bit rate in MB/hour, which reduces the retention period.

Therefore the formula to calculate the Write Rate becomes:

Write Rate (W) = (B * P * U) * (1 + K * M)

where

- K is the retained .mp4 average bit rate (in MB/hour)
- M is the proportion of recorded session hours which were converted to .mp4

Clients are now encouraged to fetch .mp4 and .wav files using the wavUrl and mp4url links directly, rather than use the convertSession API. Using these links, MediaSense performs the conversions on demand, resulting in a single-step download procedure. The converted files are retained for a period of time in case a second request is received, but then they are cleaned up automatically. Therefore, you do not need to consider these files in your disk space calculations.

There is also an absolute maximum number of recordings that MediaSense can retain, no matter how much disk space is provisioned or the length of the recordings. That maximum depends on the number of tags, tracks, participants, and other metadata elements per recording; but it is generally about 16 million recording sessions.

CUBE capacity

A Cisco 3945E ISR G2 router, when running as a border element and supporting simple call flows, has a capacity of about 1000 simultaneous calls (if equipped with at least 2 GB—preferably 4 GB of memory). In many circumstances, with multiple call movements, the capacity will be lower—in the range of 800 calls (due to the additional signaling overhead). In addition, the capacity will further be reduced when other ISR G2 functions (such as QoS, SNMP polling, or T1 based routing) are enabled.

Some customers will need to deploy multiple ISR G2 routers in order to handle the required call capacity. A single MediaSense cluster can handle recordings from any number of ISR G2 routers.

Network bandwidth provisioning

For Call Recording

If Call Admission Control (CAC) is enabled, Unified Communications Manager automatically estimates whether there is enough available bandwidth between the forking device and the recording server so that media quality for either the current recording or for any other media channel along that path is not impacted. If sufficient bandwidth does not appear to be available, then Unified Communications Manager does not record the call; however, the call itself does not get dropped. There is also no alarm raised in this scenario. The only way to determine why a call did not get recorded in this situation is to examine its logs and CDR records.

It is important to provision enough bandwidth so that this does not happen. In calculating the requirements, the Unified Communications Manager administrator must include enough bandwidth for 2 two-way media streams, even though the reverse direction of each stream is not actually being used.

Bandwidth requirements also depend on the codecs in use and, in the case of video, on the frame rate, resolution, and dimensions of the image.

For Video Playback

Media connection negotiation is still bi-directional for video playback (even though MediaSense only sends data and does not receive it). This is an important consideration since the use of bi-directional media implies that you must provision double the bandwidth than what you might have otherwise expected.

Impact on Unified Communications Manager sizing

MediaSense does not connect to any CTI engines, so the CTI scalability of Unified Communications Manager is not impacted. However, when MediaSense uses Cisco IP phone built-in-bridge recording, the Unified Communications Manager BHCA increases by 2 additional calls for each concurrent recording session.

For example, if the device busy hour call rate is six (6) without recording, then the BHCA with automatic recording enabled would be 18. To determine device BHCA with recording enabled, use this calculation:

(Normal BHCA rate + (2 * Normal BHCA rate)

For more information, see "Cisco Unified CM Silent Monitoring & Recording Overview.ppt" under SIP Trunk documents at http://developer.cisco.com/web/sip/docs.