Solution Reference Network Design for Cisco MediaSense

First Published: February 25, 2013

Last Modified: April 26, 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

Cisco MediaSense Release 9.1(1) SRND

Table of Contents

Product Overview

Cisco MediaSense is a SIP-based, network-level service whose function is to provide voice and video media recording capabilities for other network devices. Fully integrated into Cisco's Unified Communications architecture, Cisco MediaSense automatically captures and stores every Voice over IP (VOIP) conversation which traverses appropriately configured Cisco Unified Communications Manager (Unified CM) IP phones or Cisco Unified Border Element (CUBE) devices. In addition, an IP phone user or SIP endpoint device may call the Cisco MediaSense system directly in order to leave a recording consisting of only media generated by that user. Such recordings may 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, Cisco MediaSense allows you to record a conversation from different perspectives, depending on your needs. Recordings forked by an IP phone are taken from that phone's perspective: 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), because the forking phone is no longer involved. This perspective may work well for contact center supervisors whose focus is on a particular agent.

Recordings forked by CUBE on the other hand, offer 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 interaction 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 CM 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, QuickTime, 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 files and downloaded in that format. All access to recordings, either in progress or completed, is through web-friendly URIs. Cisco MediaSense also offers a web-based Search and Play application with its own 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, Cisco MediaSense offers the ability to play back specific video media files on demand. This capability is designed to support Video in Queue (ViQ), Video on Demand (VoD), or Video on Hold (VoH) use cases, in which a separate call controller invites Cisco 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 causes the video to be played by sending a SIP Invite to Cisco MediaSense at the selected dialed number.

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

Cisco MediaSense also maintains a metadata database, in which information about all recordings is maintained. A comprehensive Web 2.0 API is provided which allows client equipment to query and search the metadata in various ways, to control recordings that are in progress, to stream or download recordings, and to bulk-delete recordings which meet certain criteria, as well as to apply custom tags to individual

recording sessions. A Symmetric Web Services (SWS) eventing capability is also provided, which allows 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 Cisco 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 or VoH 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 also particularly well served by the fact that Cisco MediaSense maintains the two sides of a conversation as separate tracks and provides access to each track individually, greatly simplifying the analytics engine's need to identify who is saying what.

Characteristics and Features

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

Compliance Recording

In *compliance recording*, calls are statically configured to be recorded. For IP phone recording, all calls which are 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 which match particular dial peers (typically selected by dialed number pattern) are recorded. Cisco MediaSense does not itself control which calls are recorded (except to the limited extent described under Incoming Call Handling Rules). Compliance recording is distinguished from *selective recording*, in which the recording server would have the ability to determine which calls it will record, or even to select a new or existing call and initiate recording on it. Cisco MediaSense itself does not support selective recording, but the same effect can be had by deploying Cisco MediaSense in combination with certain partner applications.

Recording is accomplished by "media forking" – the phone or CUBE in effect sends a copy of the incoming and outgoing media streams to the Cisco MediaSense recording server. When a call originates or terminates at a recording-enabled phone, Unified CM sends a pair of SIP Invites to both the phone and the recording server. The recording server then 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 Invite to the recording server. The recording server then 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 which flows in each direction. These two streams are captured separately on the recorder, though both streams (or "tracks") are guaranteed to end up on the same Cisco MediaSense recording server. (Further implications of this dual track architecture are discussed below.)
- Most, but not all Cisco IP phones support media forking (a list is provided below). 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 will be the codec used in recording. Cisco MediaSense supports a limited set of codecs; if the phone negotiates a codec which is not supported by Cisco 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, and could take some time to complete. There is therefore a possibility of clipping at the beginning of each call. Clipping is typically limited to less than two seconds however, but it can be affected by overall CUBE, Unified CM, and Cisco MediaSense load, as well as by network performance characteristics along the signaling link between CUBE or Unified CM and Cisco MediaSense. Cisco MediaSense carefully monitors this latency and raises alarms if it exceeds certain thresholds.

As mentioned above, Cisco MediaSense does not itself initiate compliance recording; it only receives SIP Invites from Unified CM or CUBE, and is not involved in deciding which calls should or should not be recorded. The IP phone configuration and/or the CUBE dial peer configuration determines based only on its own configuration whether media should be recorded. This gives rise to the possibility that some calls may be multiply recorded, with neither CUBE, Unified CM nor Cisco MediaSense being aware of that fact while it is happening. Such 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 itself could actually end up creating two recordings of its own (we'll discuss this later). That said, Cisco MediaSense stores enough metadata that a client can invoke a query to locate duplicate calls, and selectively delete one copy if so desired.

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

Finally, Cisco MediaSense can record calls up to eight hours in duration.

Transfers and Conferences

Cisco MediaSense recordings are made up of one or more *sessions*. As described above, each media forking session contains two media streams, one for the incoming stream and one for the outgoing stream. A simple scenario consisting of a straightforward two-party conversation is represented entirely by a single session. A multi-party conference is no different: there is still only one stream in either direction, with the conference bridge itself combining all but one of the parties into a single Cisco MediaSense participant. In the case of phone forked media, there is an indication in the metadata that one of the streams represents a conference bridge, but in no case does Cisco MediaSense have the full list of conferenced parties.

Transfers behave somewhat differently depending on whether the call is forked from a Unified CM phone or from a CUBE. With Unified CM recordings, the forking phone itself anchors the recording. Transfers which end up dropping the forking phone will terminate the recording session, whereas transfers which 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. Therefore, transfers on either side of the device are usually accommodated within the same recording session. (See Solution Level Deployment Models for more information.)

When sessions included transfer and conference activities, Cisco MediaSense does its best to retain the related information in its metadata. Except when conferences are involved, the metadata keeps track of which participants are recorded in which track of the session, as well as when they entered and exited the conversation. If a recording gets divided into multiple sessions, metadata is also available to help client applications correlate those sessions together.

Hold and Pause

These two concepts sound similar, but they are not the same:

- Hold/Resume takes place as a result of a user pressing a key on his phone. Cisco MediaSense is a passive observer;
- Pause/Resume takes place as a result of a client application issuing a Cisco MediaSense API request to temporarily stop recording while the conversation continues.

Hold/Resume behavior differs depending on which device is forking media. In Unified CM deployments, one party places the call on hold, blocking all media to or from that party's phone. The other phone typically receives music (MOH). If the forking phone is the one which invokes the hold operation, then Unified CM actually 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. However, this behavior has implications for live monitoring, since the call being monitored is identified by its recording sessionId. Once the forking phone goes on hold, the current recording session terminates and the live monitoring listener will hear nothing more, even after the phone call resumes, because it will be resumed under a new sessionId.

If the forking phone is not the one which invokes the hold operation, then 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 CUBE deployments, hold and resume are implemented as direct SIP operations, and the SIP protocol itself has no direct concept of hold and resume. Instead, these operations are implemented in terms of media stream inactivity events. Cisco MediaSense captures these events in its metadata and makes it available to application clients if required, but the recording session itself 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 Cisco MediaSense API client in order to temporarily stop recording. The result is similar to what happens when you press the Pause button while recording on a VCR. The subsequent playback will simply skip over the paused segment. Cisco MediaSense does however store the information in its metadata, and makes it available to application clients if required.

The Pause feature behaves identically for CUBE and Unified CM recording.

Direct Inbound Recording

In addition to Compliance Recording, controlled by a CUBE or Unified CM Recording Profile, recordings can be initiated by directly dialing a number which is associated with Cisco MediaSense, and configured in Cisco MediaSense for automatic recording. Such recordings are not carried out through media forking technology, and are therefore not limited to CUBE or Cisco IP phones, and they are not limited to audio media. This is in fact how the Video Blogging use case is accomplished. Even a non-IP phone such as a home phone can be recorded, if it is converted to IP using a supported codec, via some sort of gateway and delivered to Cisco MediaSense.

Direct Outbound Recording

Using the Cisco MediaSense API, it is possible for a client to request Cisco MediaSense to call a phone number. When the recipient answers, his call will be recorded just as if he had dialed the recording server as in direct Inbound Recording. The client can be any device capable of issuing an HTTP request to Cisco MediaSense; for example a button on a web page could cause Cisco MediaSense to call the user's phone, at which point the system would begin recording his message. Any phone, even a non-IP phone such as a home phone can be recorded, if it is converted to IP using a supported codec, via some sort of gateway and delivered to Cisco MediaSense. Supported IP video phones can also be recorded in this way.

Direct Outbound Recording is only supported if Cisco MediaSense can reach the target phone number through a Unified CM system. In

CUBE-only deployments where Unified CM is not used for call handling, Direct Outbound Recording is not supported.

Monitoring

While a recording is in progress, Cisco MediaSense allows the session to be *monitored* through a third party streaming media player, or through Cisco MediaSense's built-in media player. In order to monitor a call from a third party streaming media player, a client must specify a Real Time Streaming Protocol (RTSP) URI, and it must be prepared to supply HTTP-BASIC credentials, and it must be capable of handling a 302 redirect. It can obtain that URI either by querying the metadata for it, or by capturing session events. Cisco MediaSense offers an HTTP query API which 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, whereby Cisco MediaSense will send it Symmetric Web Service (SWS) events whenever a recording is started (among other conditions). In both cases, 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, QuickTime, and RealPlayer. Each of these has tradeoffs however, which should be taken into account when selecting which one to use. Recall that recording sessions are usually made up of two audio tracks. Cisco 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. On the other hand, VLC is open source, and is easy to embed into a browser page. QuickTime and RealPlayer can play the two streams as stereo – one stream in each ear – but their buffering algorithms for slow connections sometimes result 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. Also, none of these players can render AAC, g.729 or g.722 audio. A custom application must be created in order to monitor or play streams in those forms.

Cisco MediaSense's built-in media player can currently only be accessed though its built-in Search and Play application, discussed below. This player covers more codecs and can play both streams simultaneously, but it still has limitations due to dependencies on the underlying operating system. In particular, when running the Search and Play application through a web browser which is running on Microsoft Windows, only AAC, g.711 and g.729 codecs are available. When running it through a browser which is running on an Apple Macintosh, only AAC, g.711 and g.722 codecs are available. This applies to both playback of recorded calls and monitoring of active calls.

And finally, keep in mind that only calls which are being recorded are available to be monitored. Customers who require live monitoring of unrecorded calls, or who cannot live with these other restrictions, may wish to consider Unified CM's Silent Monitoring capability instead, as driven through a contact center or other CTI application.

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 (see discussion above). 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. Different representations are suitable for different purposes. Via a set of custom header parameters on the RTSP PLAY command, Cisco MediaSense allows the requesting client to specify one of the following:

1. 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. RTP stream does not pause, timestamp reflects the fact that there was no pause, but RTP packets contain "TIME" padding which includes the absolute UTC time at which the packet was recorded;
3. RTP stream compresses the silent period to roughly half a second, and in all other respects acts exactly like bullet #1 above. This is the default behavior, and this is what will be observed when using the built-in media player.

In all cases, note that the file duration as 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. Furthermore, the session duration as returned by the Cisco MediaSense API and session events may also differ, since these are based on SIP activity rather than on media streaming activity.

Commercial media players such as VLC, QuickTime, and RealPlayer will always end up eliciting the default behavior, which is #3 above. However, these players are designed to play music and podcasts; they are not designed to handle media streams which include silence. They will therefore evidence various ill effects, such as hanging, disconnecting, or an inability to properly seek backward and forward in the stream.

Conversion and Download

Completed recording sessions can also be converted on demand to .mp4 format via an HTTP request. Files in this format can actually carry two audio tracks not as a mixed stream, but as stereo. Alternatively, .mp4 files can also carry one audio and one video track. Once converted, .mp4 files remain in Cisco MediaSense's storage along with their raw counterparts, and are accessible via their own URLs. As with streaming, browser or server-based clients may obtain these URIs by either querying the metadata or monitoring recording events. The URI may then be

invoked by the client for playing and/or for downloading. As with RTSP streaming, the client must provide HTTP-BASIC credentials, and be prepared to handle a 302 redirect. .Mp4 conversion therefore offers a secure, convenient and standards-compliant way to package and export recorded sessions, either for a more natural listening experience, or for long term archiving purposes.

Large scale conversion to .mp4 would take a fair amount of processing power on the recording server however, 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, Cisco MediaSense also offers a "low overhead" download capability. This allows clients, again using specific URIs, to download individual tracks in their raw g.722, g.711 or g.729 format, unmixed and unpackaged. The transport is HTTP 1.1 Chunked, which leaves it up to the client to reconstitute and package the media into whatever format best meets its requirements, and a fair amount of programming expertise will be required. As with the other retrieval methods, the client must provide HTTP-BASIC credentials, and be prepared to handle a 302 redirect. Note that AAC-LD encoded audio streams cannot currently be downloaded in this way.

Embedded Search and Play Application

Full scale deployments of Cisco MediaSense normally require a partner application for finding and playing recordings, as well as for interacting with more advanced capabilities such as quality management and voice analytics. For less sophisticated deployments, as well as for lab use, Cisco MediaSense offers its own built-in Search and Play application.

This is a web-based tool which offers the ability to search 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 format, or played using the application's built-in media player. Live streams can also be monitored in the same way.

Access to the Search and Play application is controlled using standard API user credentials.

This is an introductory version of the tool, and you may find its capabilities somewhat limited. For example, combination searching (searching by two or more criteria at the same time) is not yet supported, and no query can return more than 100 rows. As a result there are some cases where you will not be able to find a particular recording. On the other hand, it is only a straightforward client of the standard Cisco MediaSense API, and similar source code is published on the Cisco Developer Network (CDN). Extending the application for your own use may be a palatable alternative to engaging a partner application.

Embedded Streaming Media Player

Phone recording uses a different set of codecs than are 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 Cisco MediaSense records. Partner applications generally provide their own media players for this reason, as does the Cisco MediaSense built-in Search and Play application. Currently the specific codecs supported by the embedded player are somewhat dependent on the operating system on which the web browser is running. When running on Microsoft Windows, only AAC, g.711 and g.729 codecs are available. When running on an Apple Macintosh, only AAC, g.711 and g.722 codecs are available. This applies to both playback of recorded calls and monitoring of active calls.

The embedded media player is currently only available through the Search and Play application.

Uploaded Videos for ViQ, VoD and VoH

To support Video in Queue, Video on Demand, and Video on Hold use cases, Cisco MediaSense provides the ability for administrators to upload MP4 video files for later playback on demand. There are three steps to the process:

1. Produce an MP4 video which meets the technical specifications below
2. Upload the MP4 video into the Cisco MediaSense Primary node. The video automatically gets converted into a form which can be played back to a supported video endpoint, and distributed to all other nodes. Playback is then automatically load balanced across the cluster
3. Create an "incoming call handling rule" which 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 steps 2 and 3 above. These functions are not available through the Cisco MediaSense API.

MP4 is a container that may contain many different content formats. Cisco MediaSense requires that the file content meet the following specifications:

- Must contain one audio track and one video track
- Video must be encoded using H.264 Constrained Baseline Profile
- Video resolution should be 1080p, 720p, 480p or 360p, all at a 4:3 aspect ratio
- Audio must be encoded using AAC-LC
- Audio sampling frequency must be 48000 Hz
- Audio must be monaural
- The entire MP4 file size must not exceed 2GB, though the actual duration is not limited.

The above information is known as the Studio Specification. It should be provided to any professional studio which is producing video content for this purpose.

The video resolutions and aspect ratio mentioned above are actually not a requirement of Cisco MediaSense. Since Cisco MediaSense will play back exactly the resolution that it finds in the uploaded file, it is important to use the greatest resolution which looks good on all the endpoints on which you expect the video to be played. Also, the endpoints do not support AAC-LC audio, which is the standard for MP4. As a result, Cisco MediaSense automatically converts the audio to AAC-LD. However, only the EX-60 and EX-90 telepresence endpoints currently support AAC-LD, so this entire feature is effectively limited to use on those devices.

Finally, note that video playback is not supported in Services Ready Engine (SRE) deployments.

Cisco MediaSense comes with a sample video, preloaded and preconfigured, and available to use out of the box. After successful installation or upgrade, you can dial the SIP URL **sip:SampleVideo@<mediasense-hostname>** from your supported EX-series telepresence endpoint for a test run.

Incoming Call Handling Rules

When Cisco 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 should be taken, and what the default action should be. 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 Cisco MediaSense. If your application is to record calls forked by a Unified CM 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 for Record by default.

Codecs Supported

Cisco MediaSense can accept audio in g.711 ulaw/alaw, g.722, or g.729, g.729a, g.729b, and video in h.264 encoding. Note that off-the-shelf streaming media players typically do not support the g.722 and g.729 codecs, though the media player which is embedded in the built-in Search and Play application can support these codecs under certain conditions. Uploaded videos must be provided in MP4 format, using h.264 for video and AAC-LC for audio (see the exact Studio Specification below), but the audio is converted to AAC-LD for streaming playback. Most media players (including the built-in one) as well as Cisco EX-60 and EX-90 telepresence endpoints can play this format.

Neither Unified CM nor CUBE performs a full codec negotiation with Cisco MediaSense. They first negotiate codecs among the conversation endpoints, and only then initiate a connection to Cisco MediaSense. If they happen to select a codec which is not supported by Cisco MediaSense, then the call will not be recorded. Therefore, for all phones that need to be recorded, it is important to configure them such that the codec that gets selected for the phones can only be one of the codecs that Cisco MediaSense supports.

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

Cisco MediaSense is also capable of recording the audio portion of Cisco Telepresence calls, among EX-60, EX-90, and CTS-500 devices. However, these endpoints must be configured to use a g.711 or g.722 codec.

Mid-call codec changes may be called for based on call flow activities, most notably when the call is transferred or conferenced with a phone which has different codec requirements than those which were negotiated during the initial Invite. The results differ however, depending on whether CUBE or Unified CM is providing the forked media. With Unified CM 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 CM will automatically try to insert a transcoder between the two phones so that the recording codec can remain constant. If no transcoder is available, Unified CM will drop the transferred call and terminate the recording.

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

One additional note: it is not possible for the two audio tracks in a session to be assigned different codecs. Neither CUBE nor Unified CM can create this situation.

Metadata Database and the Cisco MediaSense API

Cisco 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, among other things:

- 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 above information, Cisco MediaSense also stores *tags* with 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. Untimed 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.

Cisco MediaSense itself 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 a session, media inactivity state change tags are further associated with a specific track in the session.

Cisco MediaSense API

The Cisco MediaSense API offers a number of methods to search and retrieve information in the metadata database. Suitably authenticated clients may perform simple lookups, such as finding all sessions which have been deleted by the automatic pruning mechanism, or all sessions within a particular time range which 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 Cisco MediaSense functions as well. Clients can use the API to subscribe for events, to manage disk storage, to manipulate recording sessions which are in progress, to remove unneeded inactive sessions and recover their resources, and to invoke operations such as conversion to .mp4. Lengthy operations are supported through a remote batch job control facility. The API is described in detail in the Cisco MediaSense Developer Guide.

Cisco 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, though the client needs to authenticate to each server separately, and to provide the HTTP session identifier which was obtained from the server being addressed.

Events

The Cisco MediaSense eventing mechanism is designed to provide 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 all of the critical information about a session given its current state, which a client could use to offer various interesting and powerful applications. A client could for example, use the URIs provided in these events in order to offer real time monitoring and control buttons to an auditor or contact center supervisor. It might also implement a form of *selective* recording (as opposed to *compliance* recording), by deleting, after the fact, sessions which it decides did not need to be recorded.

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

Storage Threshold events are useful in allowing a server-based client application to manage disk usage. The client would typically subscribe for these events, and selectively delete older recordings when necessary, according to its own rules. For example, during the normal course of operations, the client might use the tagging facility to mark selected sessions for retention, and then when a threshold event is received, delete all sessions older than a certain date, but skipping over those that have been 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 Cisco MediaSense to the client (note that HTTPS is *not* currently supported for eventing). When a client subscribes, it provides a URL to which Cisco 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. This has implications for high availability which will be discussed below; for now suffice it to say that customers must choose one of two modes of event delivery – one which favors reliability, and one which favors convenience.

Metadata Differences between CUBE and Unified CM

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

First, there is the essential difference in topology: with Unified CM, calls are forked by one of the endpoints, whereas 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 CM they are different devices. These lead to differences in the way participants are identified and in the way they are associated with media tracks.

But the more substantial deviations have to do with mid-call activities. For example, Hold and Resume trigger a new session for Unified CM calls, but they instead insert track level tags for CUBE calls. Transfer of a forking phone terminates a recording session, whereas such an action is not even possible with CUBE calls. Conference detection varies considerably: with Unified CM it appears as a transfer to a conference bridge, with updated participants, and a metadata flag identifies it as a conference; with CUBE it also appears as a transfer, but the metadata flag is not supported, and the participant data is erratic and unreliable.

There are significant differences with respect to call correlation as well. Unified CM and CUBE use different mechanisms for identifying calls, and evidence of those differences does filter into Cisco MediaSense.

The bottom line is that 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 CM.

The Cisco MediaSense Developers Guide contains a full list of differences between CUBE and Unified CM deployments.

Disk Space Management

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

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

- Recording Priority
- Retention Priority

Recording Priority mode is designed for customers who would rather lose an old recording than miss a new one. In this mode, Cisco MediaSense automatically prunes recordings which 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. In this mode Cisco MediaSense will not automatically prune recordings for any reason. In either mode, Cisco MediaSense will stop accepting new calls if necessary in order to protect the space remaining for calls which are currently in progress. Affected calls will be automatically redirected to another Cisco MediaSense recording server if one is available.

The algorithms are as follows:

Retention Priority behavior

- No automatic pruning takes place
- When a server enters warning condition (75%) an alarm will be raised
- When a server enters critical condition (90%) it will redirect new calls
- When a server enters emergency condition (99%) it will drop active recordings
- When a server exits critical condition (drops below 87%) it will start 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 warning condition (75%) an alarm will be raised
- When a server reaches critical condition (90%) older recordings (even if younger than the age threshold) will be pruned to make room for new ones
- When server enters emergency condition (99%) it will redirect new calls and forcibly drop ongoing recordings
- When a server exits the emergency condition (drops below 97%) it will start taking new calls

Any automatic pruning applies only to raw recording files. An administrative option determines whether Cisco MediaSense should then automatically delete any converted .mp4 recordings, as well as any metadata associated with pruned recordings. If not enabled, then an API client *must* take responsibility for deleting them.

Clients therefore have the option of managing disk usage directly. Cisco MediaSense takes progressively more aggressive action when storage levels reach successively 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, and the Cisco MediaSense API offers a number of ways in which the client can choose which recordings should be deleted, including an option to issue a customized bulk delete operation which is then carried out without client involvement.

The ability to explicitly delete old recorded sessions is not limited to automatic operations performed by a server based client. A customer may wish to take a completely manual approach, designing a web page which fetches and displays appropriate meta-information about older recordings, and allows an administrator to selectively delete those which he considers to be expendable. Such a web page would use the same API as the server-based client would use.

System Resiliency and Overload Throttling

Cisco 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. New recordings when the node is at capacity will be redirected to other nodes if available; if not then they will be rejected and lost. Media output requests -- live monitoring, playback, raw download and mp4 conversion -- will result in 503 responses when the node is at capacity. The relative weight of various media is also considered in this algorithm; video takes significantly more capacity than audio. Since recording is always considered to be a higher priority operation, Cisco 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.

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 Cisco 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 deployment such that the overall rated node and cluster capacities are not exceeded.** Capacity specifications can be found in the Scalability and Sizing section below.

Cisco MediaSense also protects itself with respect to media storage capacity. As detailed in the "Disk Space Management" section above, 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 the usage levels and throttling activities of each node.

Cisco MediaSense-Specific Deployment Models

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

Server Models Supported

Cisco MediaSense may only be deployed on top of 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, on a Services-Ready Engine (SRE-910), or on selected Hewlett Packard (HP) or IBM servers. (see the Compatibility Matrix section below for versions and model numbers).

When ordering C-series servers, be sure to include battery backed cache. Their RAID controllers have an optional battery backup for the write cache. If the battery backup 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 the Compatibility Matrix section for detailed disk requirements.)

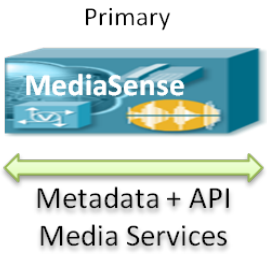
Finally, the Primary and Secondary servers (see below) *must* be based on identical hardware, or at least have identical specifications in terms of CPU speed and disk I/O throughput. They must also be using the same version of VMware ESXi. Any asymmetry will cause accumulating database latency in one server or the other. Expansion servers do not need to be running on the identical hardware.

Server Types

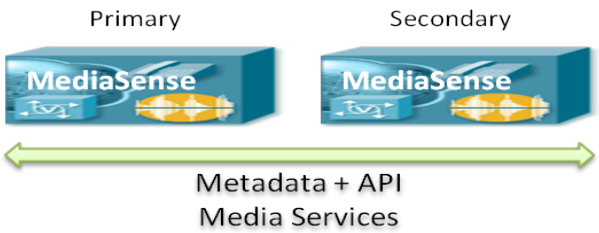
Cisco MediaSense is deployed on up to five rack-mounted servers, or two Services-Ready Engine (SRE) modules, depending on the capacity and degree of redundancy required. (For the purposes of this discussion, "server" refers to a virtual machine, not necessarily a physical machine.) There are three types of servers:

- Primary: Supports all database operations as well as media operations.
- Secondary: Provides high-availability for the database. Also supports all database operations as well as media operations.
- Expansion: Provides additional capacity for media operations, but not for database operations. Expansion servers are not used in SRE-based deployments.

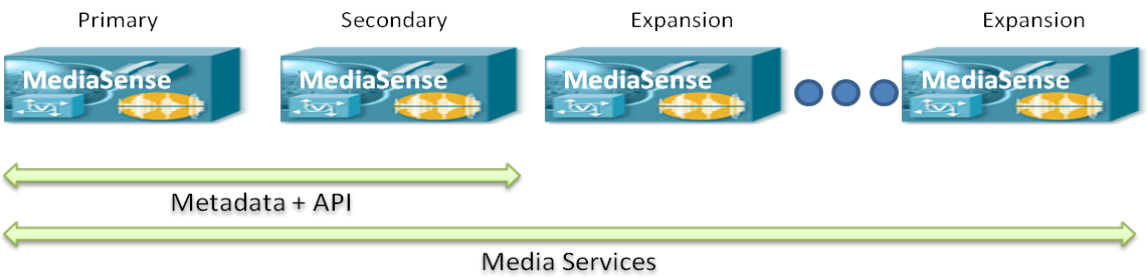
Only the Primary server is required, and indeed a small lab setup would likely consist of only that server. The following diagram depicts this deployment model:



Customers who require database redundancy would then deploy a Secondary server as well, as shown below:



If additional recording capacity is required, Expansion servers would be deployed, as follows:



Note that Expansion servers are not supported in SRE deployments.

All servers, including SRE-based servers, are provided with 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 server are identified during the initial web-based setup process for those nodes, after their installation has completed.

Each server adds both simultaneous activity capacity – the ability to perform more parallel recording, monitoring, download and playback activities – and storage capacity. Recordings are always stored on the disks which are attached to the server which initially captured the media. Servers may be added to a cluster at any time, but there is no ability to remove servers from a cluster. There is also no ability to add or remove storage capacity to a node once it has been installed.

On non-SRE two-server deployments it is possible to replace the Secondary server with an Expansion server. However, there is little reason to do so, except perhaps in lab environments. The amount of recording capacity to be gained at the expense of database redundancy is minimal, and usually does not outweigh the resulting loss of data safety. Beyond two servers, both a Primary and a Secondary server are required.

SRE-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. Note that a Cisco MediaSense cluster must be SRE-based or rack-mount server based. It cannot be made up of a combination of the two.

Very Large Deployments

Customers who require capacity which exceeds that of a single Cisco MediaSense cluster must deploy multiple independent Cisco MediaSense clusters. In such deployments, there is absolutely no interaction across clusters; no cluster is aware of the others. API clients need to be designed such that they can access all of the deployed clusters independently; some Cisco MediaSense partner applications already have this capability.

For Unified CM phone recording, phones should be partitioned among Cisco MediaSense clusters, such that any given phone will be recorded by one specific Cisco MediaSense cluster; its recordings will never be captured by a server in another cluster.

For CUBE recording, the situation is more flexible. One or more CUBE systems may direct their forked media to one or more Cisco MediaSense clusters. That said, a client cannot search for recordings in two clusters at once, since the metadata databases are independent. It can issue queries to both clusters simultaneously, however. It can also subscribe to events from both clusters simultaneously,

and even specify that both clusters should send their events to the same SWS URL. This permits a single server application to receive all events from both clusters.

Virtual Machine Configuration

Cisco provides an OVA virtual machine template for your use, in which the required CPU and memory reservations are built in. VMWare ESXi will enforce these minimums, and ensure that other VMs do not compete with Cisco 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 Cisco MediaSense VM.

The Cisco-provided OVA template contains a dropdown list which allows you to select one of the supported VMware virtual machine template options in this release for Cisco MediaSense servers. These template options specify among other things, the memory, CPU and disk footprint for a virtual machine. A total of 3 options are provided; for each virtual machine you install, you must select the appropriate template option on which to begin your installation. It is a requirement that you use this Cisco-provided template 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 an unsupported hardware configuration.

The following table summarizes the 3 template options provided.

Template Option	Type of Server	vCPUs	Memory	Disk	CPU Reservation	Maximum total recording space supported	Notes
MediaSense SRE node	SRE Module	2	6 GB	80GB for O/S; 80GB for Database and working storage; 210GB for recordings	2200 MHz	420 GB across both nodes	(2)
MediaSense Expansion node	Expansion	7	16 GB	80GB for O/S; 80GB for working storage	10000 MHz	12 Terabytes per Expansion node. The minimum 210 GB provisioned by default.	(1,2,3)
MediaSense Primary/Secondary node	Primary and Secondary	7	16 GB	80GB for O/S; 600GB for Database and working storage	15000 MHz	12 Terabytes per Primary or Secondary node. The minimum 210 GB provisioned by default.	(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 default. Additional space may be added before or after Cisco MediaSense software installation. Once provisioned, recording space may never be reduced. Note that 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. This is dictated by the size of the database disk on the Primary and Secondary nodes.

Media Storage Alternatives

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

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

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

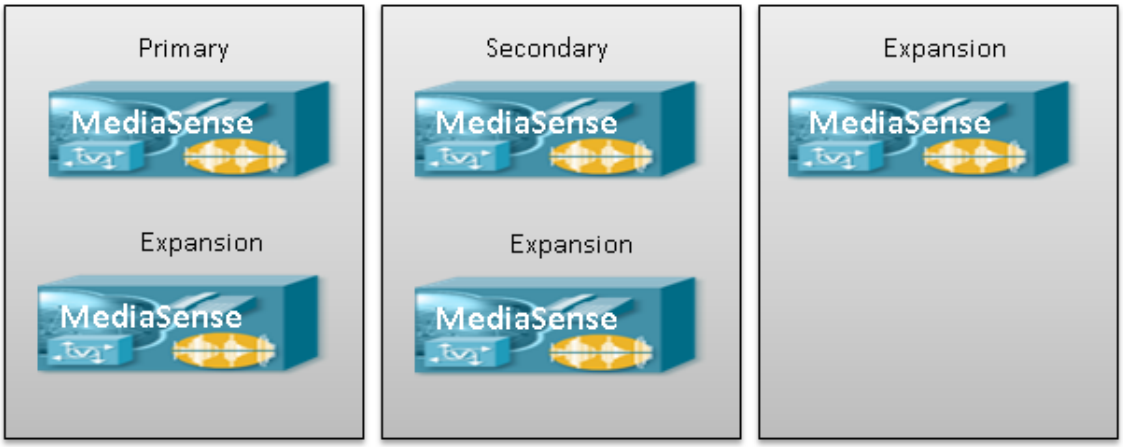
On an SRE module, there is only one storage format: each node contains exactly one physically attached 210GB media disk, for a maximum of 420GB media storage capacity across two nodes.

Deploying Multiple Cisco MediaSense Virtual Machines Per Server

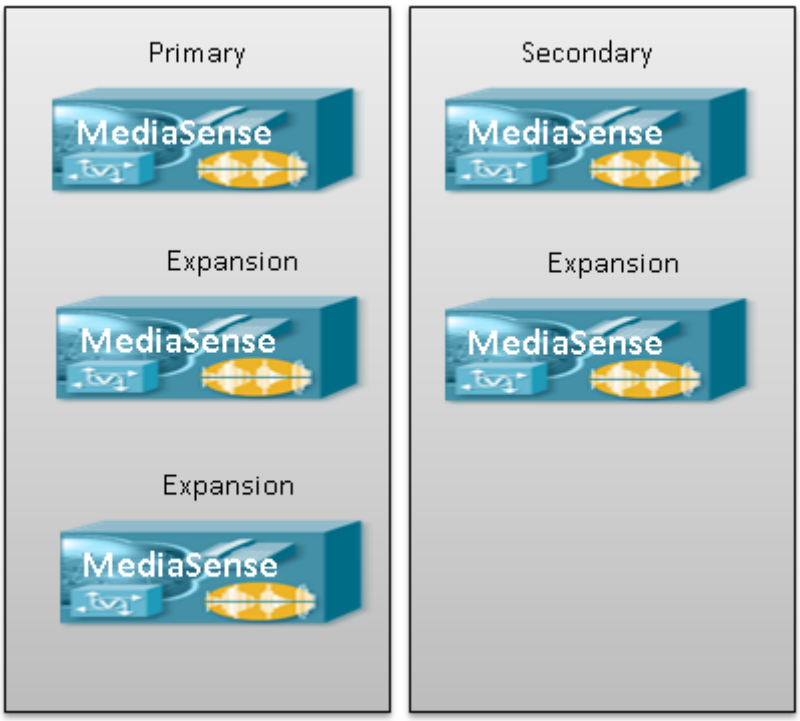
As of Release 9.0(1), Cisco MediaSense is no longer required to be the only virtual machine running on a physical server. Other applications can share the server, as long as Cisco MediaSense is able to reserve the minimum resources it requires.

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 Cisco MediaSense VMs a particular server model will support by referring to the UC Virtualization Wiki (referenced in the Compatibility Matrix section below), and use the number of physical CPU cores as a guide. Models with 8 or more physical cores can support 1 Cisco MediaSense VM; models with 14 or more physical cores can support 2 Cisco MediaSense VMs, and models with 20 or more physical cores can support 3 Cisco MediaSense VMs (a small amount of hyperthread resource is used in this case).

Geographical Specifications

All Cisco 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 Cisco 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 Cisco MediaSense cluster over a WAN, with certain potential caveats:

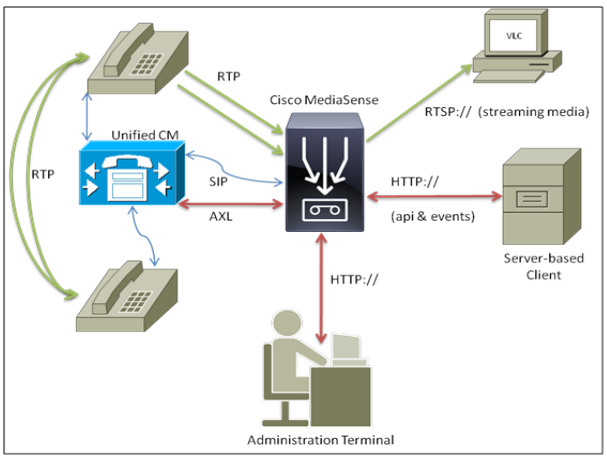
- In Unified CM deployments, media forking phones may be connected to Cisco MediaSense over a WAN.
- SIP Trunks from Unified CM may also be routed over a WAN to Cisco MediaSense, but it should be expected that affected calls may evidence additional clipping at the beginning of recordings due to the increased round trip delay.

- The connection between CUBE and Cisco MediaSense may also 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 CM and Cisco MediaSense may be routed over a WAN, but API and administrator sign-in times may be delayed.
- From a high availability standpoint, be aware that API sign-in has a dependency on the AXL link being functional. 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 HTTP Download. This applies to SRE deployments as well as centralized deployments, and to CUBE deployments as well as Unified CM deployments.

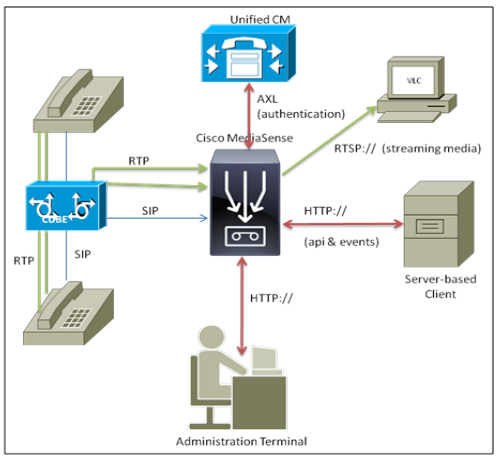
Solution Environment

Solution Overview and Call Flows

The following diagrams depict the Cisco MediaSense solution environment for Unified CM deployments and for CUBE deployments:



UCM Solution Topology



CUBE Solution Topology

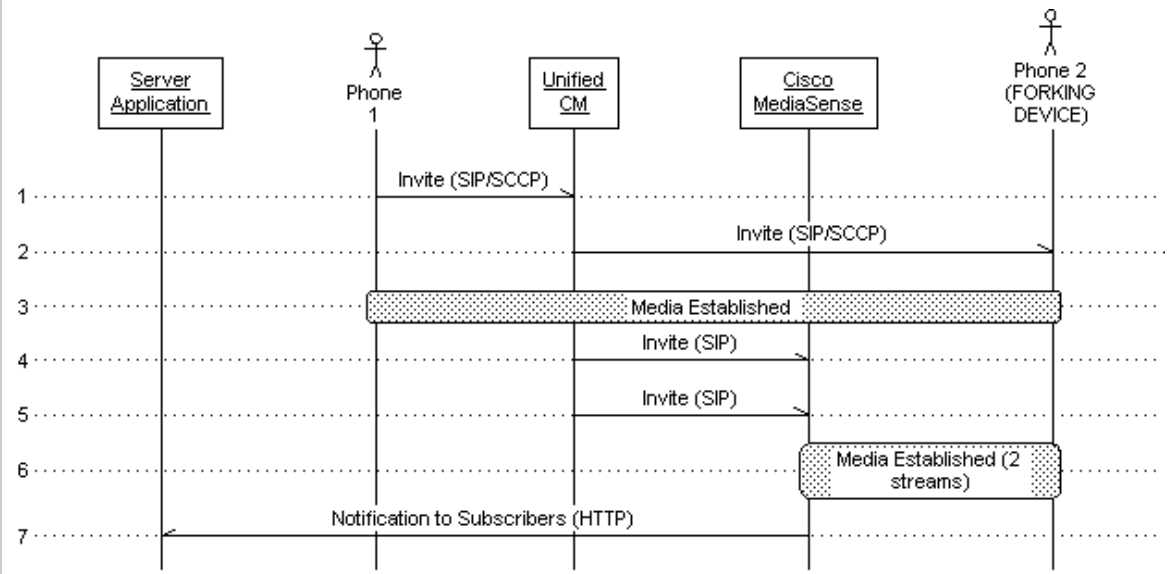
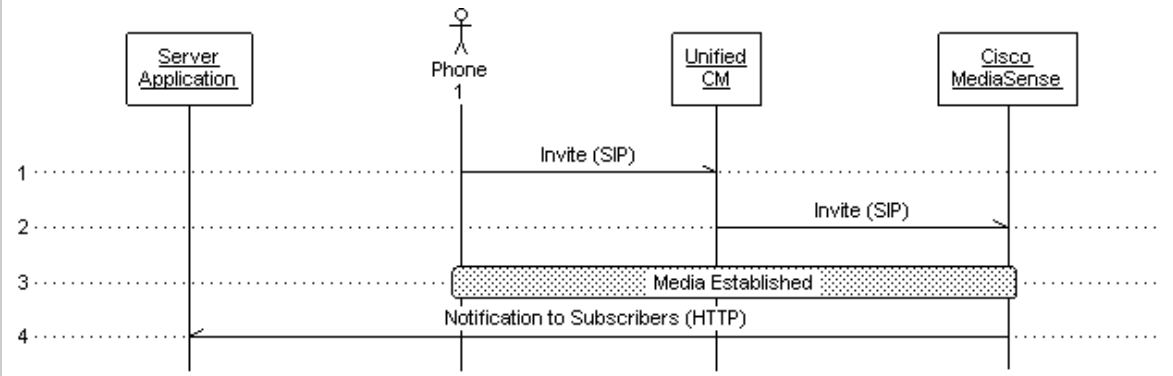
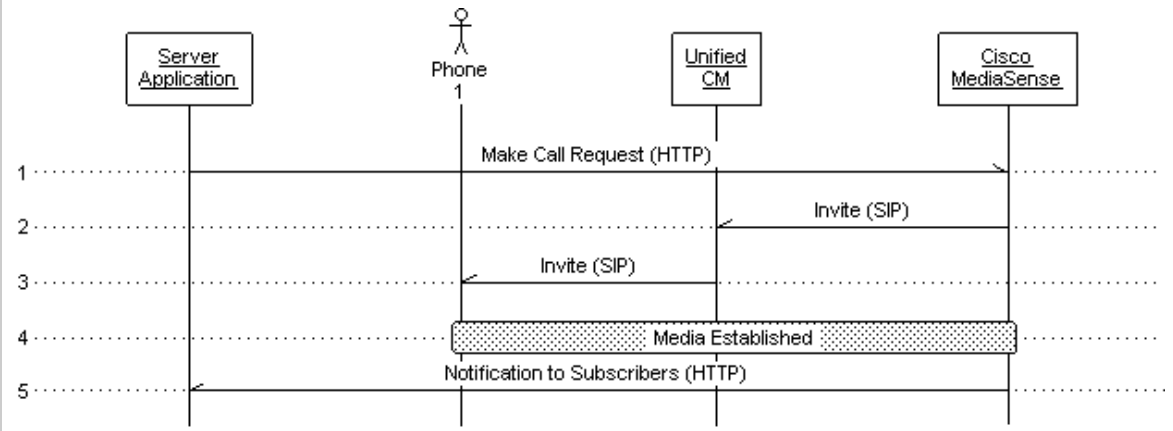
Though these diagrams each show only one Cisco MediaSense server and one Unified CM server or CUBE, each should be considered as a cluster of such devices. That is, one cluster of Cisco MediaSense servers interacts with one cluster of Unified CM servers or with one or more CUBE devices. For Unified CM deployments, there is no concept of a hierarchy of recording servers. SIP Trunks should be configured to point to all Cisco MediaSense servers. For CUBE deployments, recording dial peers should be configured to point to one or two of the Cisco MediaSense servers (preferably avoiding the Primary and Secondary). The High Availability section discusses this in more detail.

SRE deployments are built with exactly the same topology. Physically, an SRE 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. An SRE-based Cisco MediaSense cluster can even record calls which are forked from Unified CM phones.

Notice that the CUBE Solution Topology includes a Unified CM device. This is used only for authentication purposes, and has no role in call flow.

General Flow - Unified CM Calls

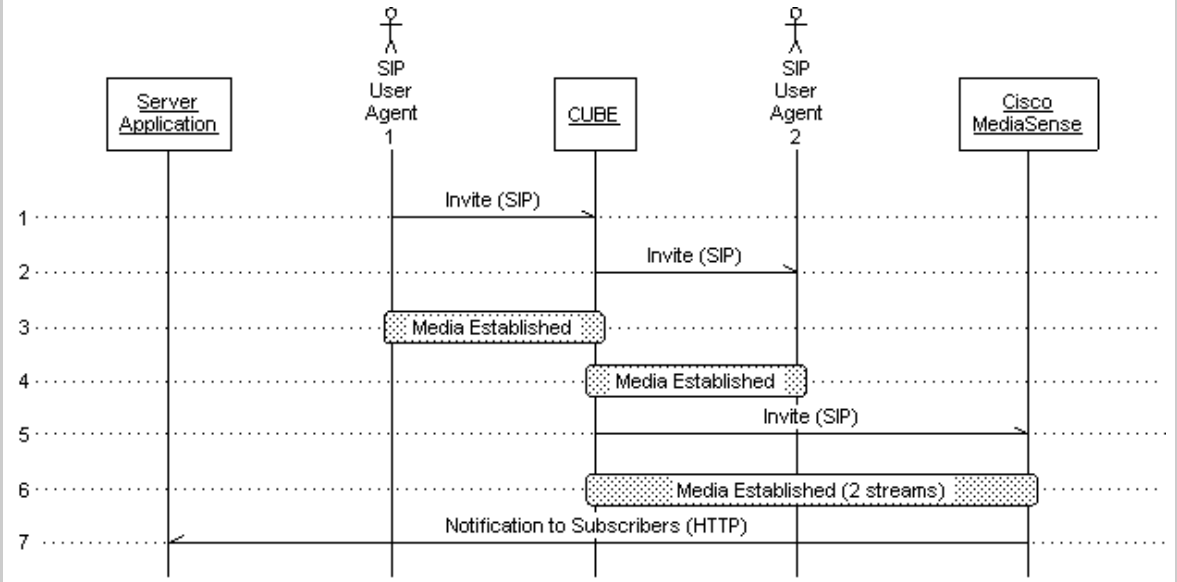
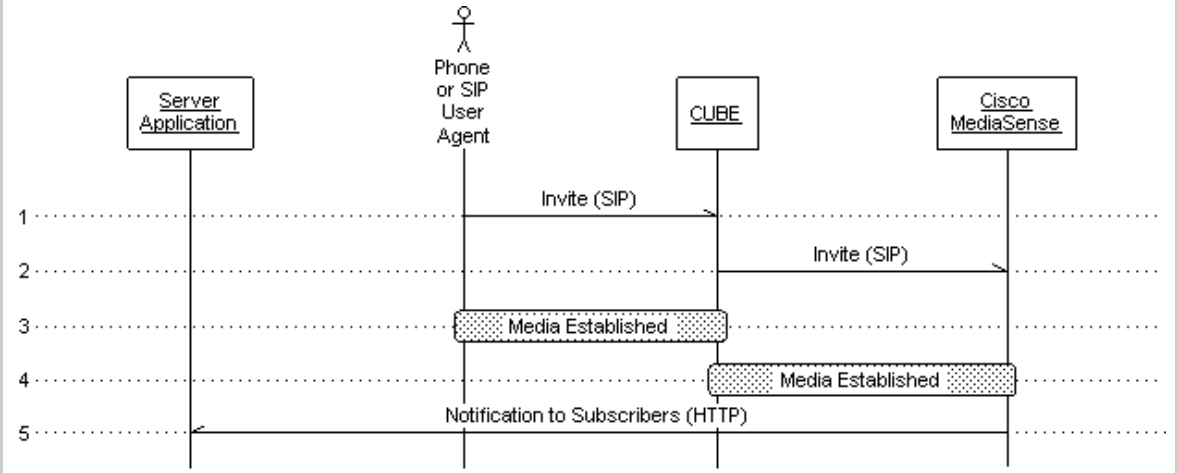
For compliance recording applications, call recordings are initiated via a pair of SIP Invites from Unified CM to Cisco MediaSense, though only after the initial call has been established between two parties. Unified CM is involved in the call setup, but media flows to Cisco MediaSense from one of the phones, not from Unified CM. Inbound blog recordings are initiated in a similar way: a SIP Invite is sent from Unified CM to Cisco MediaSense. Outbound blog recordings are initiated via an API request ("startRecording") to Cisco MediaSense, which triggers an *outbound* SIP Invite from Cisco MediaSense to Unified CM. In all cases, the processing of the Invite results in one or more RTP media streams being established between the phone being recorded and Cisco MediaSense. These call flows are depicted in the following figures (Note: these figures are illustrative only and are not intended to 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 via a *single* SIP Invite from CUBE to Cisco MediaSense, containing two "m" lines, as opposed to two separate Invites which each contain one "m" line. As with Unified CM, the Invite is sent only after the initial call has been established between two parties. However, the media which flows to Cisco MediaSense comes from CUBE, and not from one of the endpoints.

Inbound blog recordings are initiated by directly dialing a call from an endpoint, through CUBE, to Cisco MediaSense. In all cases, the processing of the Invite to Cisco MediaSense results in one or more RTP media streams being established between the CUBE and Cisco MediaSense. These call flows are depicted in the following figures (Note: these figures are illustrative only and are not intended to show the detailed message flow. Also note that *outbound* blog recordings are not supported with CUBE deployments.)

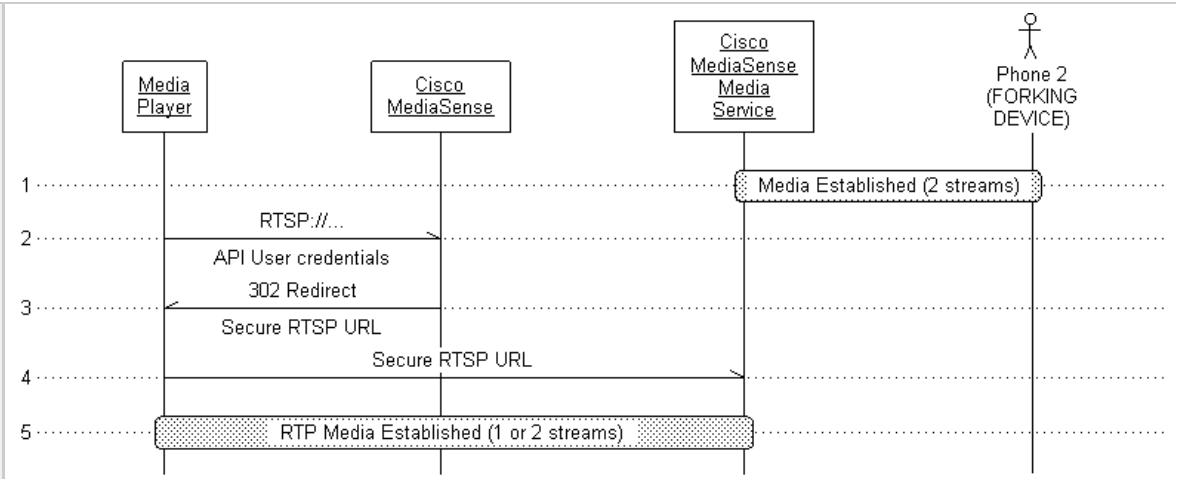
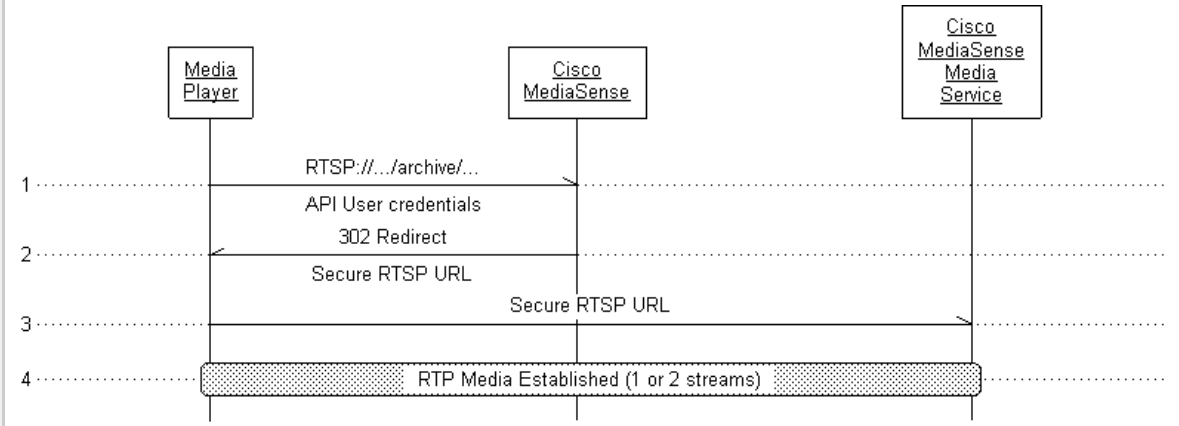
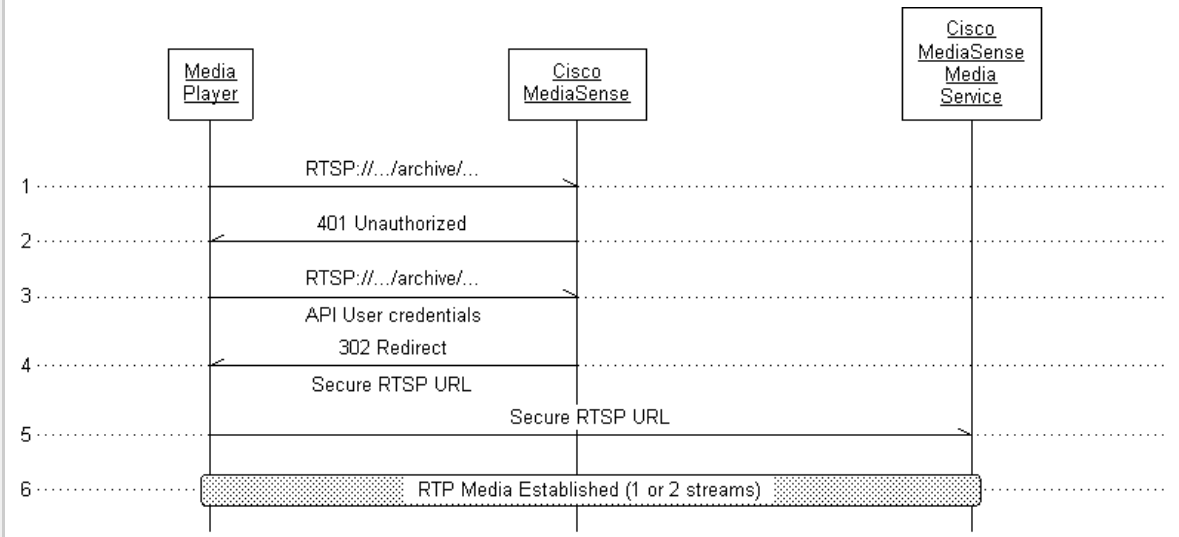
Compliance Recording for CUBE Calls	
Direct Inbound Recording for CUBE Calls	

General Flow - Streaming Media

Live monitoring is accomplished when a workstation running a streaming media player sends an RTSP:// URI to Cisco MediaSense, specifying an active media address; the RTP media stream is then established between Cisco MediaSense and the player. This stream is actually a *copy* of one of the streams that Cisco 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 Cisco MediaSense, specifying an "archive" media address. The resulting media stream between Cisco MediaSense and the player is read from the disk.

Live monitoring and playback call flows are illustrated below. (Again, these figures are illustrative only and are not intended to show the detailed message flow.) Notice that these flows also show how authentication takes place. The Cisco MediaSense Media Service is the software component within each node which is responsible for handling streaming media.

Live Monitoring	
Playback	
Playback showing Authentication Challenge	

The Cisco 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 Cisco MediaSense can be deployed as part of a solution.

The following basic models are discussed:

- I. Unified CM Deployments

- II. CUBE Deployments
- III. CUBE Deployments with Unified CVP

Finally we will cover:

- Additional options and considerations.

I. Unified CM Deployments

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

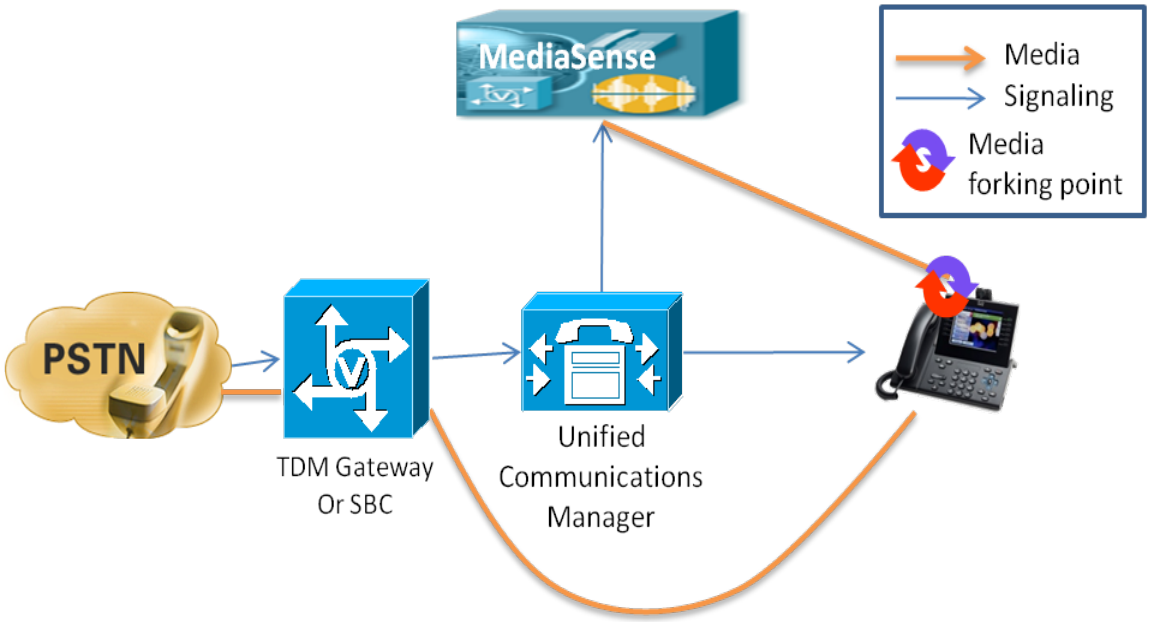
- I (a). Basic Unified CM Deployment – Internal/External
- I (b). Basic Unified CM Deployment – Internal/Internal

From the perspective of Cisco MediaSense, there is actually no difference between the two Basic Unified CM 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 they might be viewed differently at the overall solution level, and because there is a significant difference in their behavior, as explained below.

Later in this section we discuss:

- I (c). Unfied SME Deployment

I (a). Unified CM Deployment – Internal/External

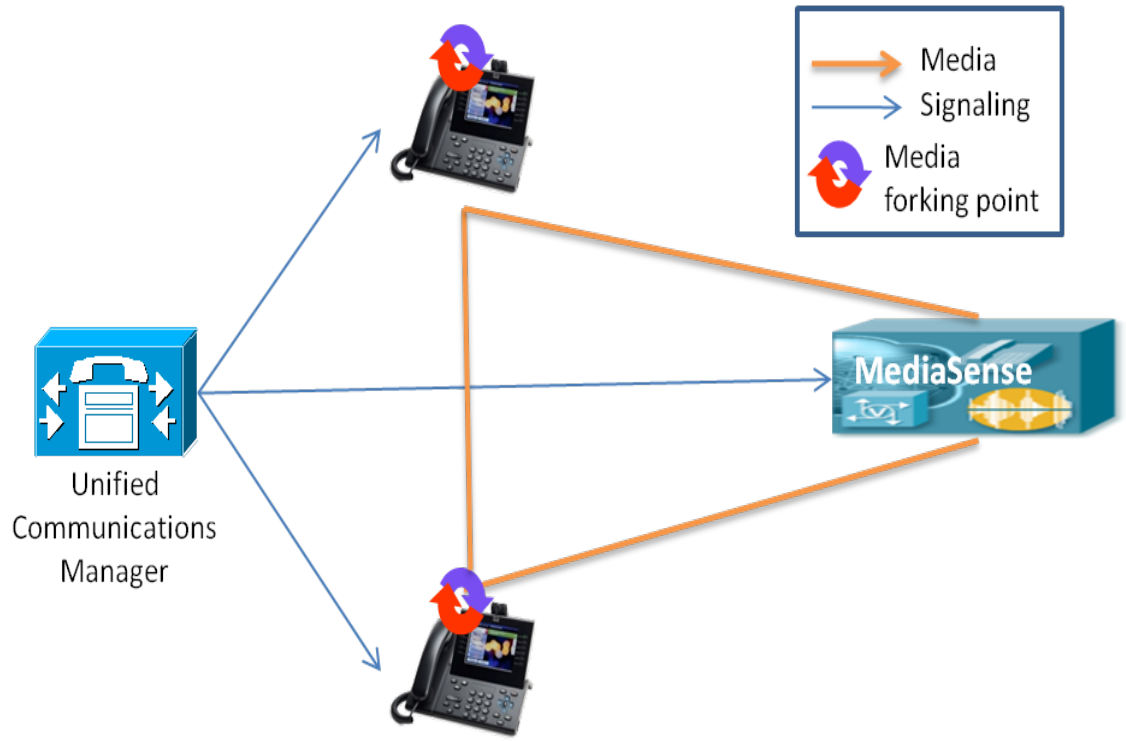


The above diagram shows a basic Unified CM deployment in which calls with parties who are outside the enterprise are recorded. This applies to both inbound and outbound calls, as long as the inside phone is configured with an appropriate recording profile.

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

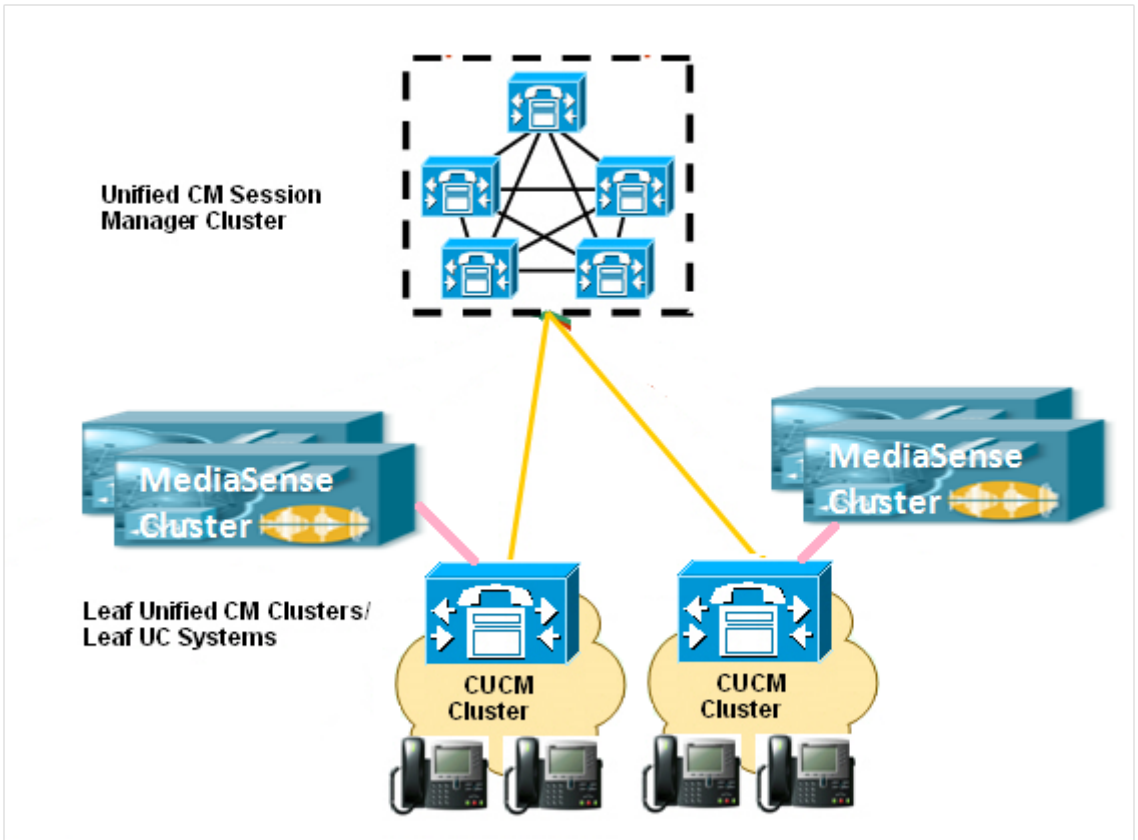
If the call is transferred away from this phone, the recording session ends. Only if the phone which takes up the call is configured for recording will the next segment of the call be captured.

I (b). Unified CM Deployment – Internal/Internal



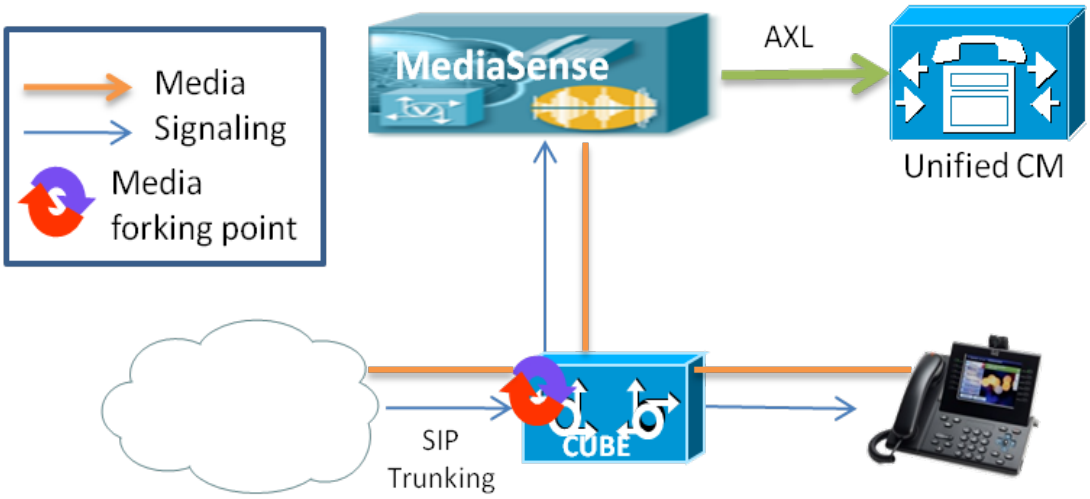
This diagram shows a basic Unified CM deployment in which calls are with parties who are inside the enterprise. Either one of the phones must be configured for recording. If both phones are configured for recording, then two separate recording sessions will be captured.

I (c). Unified SME Deployment



At this time, Cisco MediaSense does not support SME deployments in the general sense. In an SME environment, all the phones which are to be recorded by a specific Cisco 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 Cisco MediaSense clusters must be deployed, as shown in the above diagram.

II. Basic CUBE Deployment



The preceding diagram shows a very basic CUBE deployment, in which calls arrive via 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 via a recorder profile configuration which is attached to one or more dial peers.

First a note about terminology. When any 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's 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 will be the one which represents the outside of the enterprise, while the outbound dial peer will be the one which represents the inside of the enterprise. The assignment is reversed in the case of an outbound call. Thus for the purposes of this document, we use the terms *inside* and *outside* dial peers, to represent the inside and the outside of the enterprise, respectively.

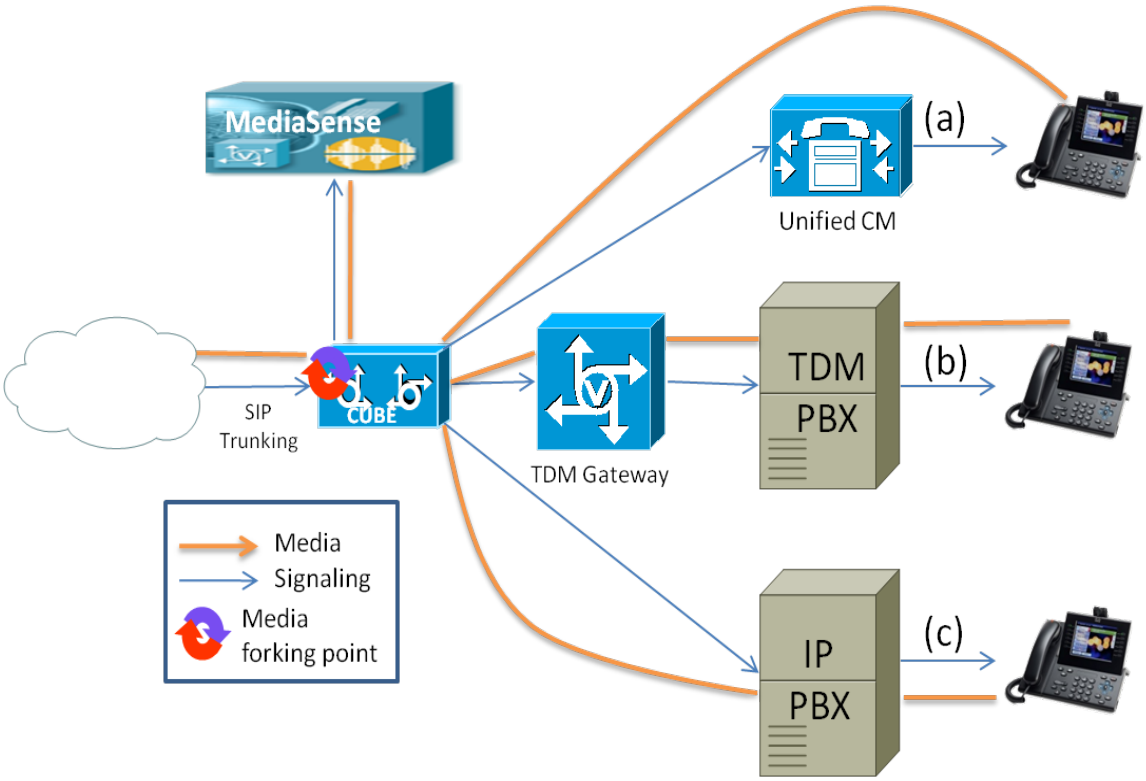
Though there are a few exceptions, best practice is to apply the recording profile to the *outside* dial peer - the inbound dial peer for inbound calls, and the outbound dial peer for outbound calls. This is because the external leg of the call is typically quite stable, whereas the internal leg is often subject to complex call manipulations, including various kinds of consults, conferences and transfers. If any of those operations, intentionally or not, cause the CUBE to trigger a new dial peer match, the recording session may be terminated prematurely.

This diagram also shows a Unified CM component. Though currently required for CUBE deployments, Unified CM does not perform any call signaling, media or record keeping. A single Unified CM server is required specifically for the purpose of managing and authenticating Cisco MediaSense API users. It can be any existing or specially installed Unified CM server, Release 8.5(1) or later. Ideally, the server selected should be one which is not itself loaded with calls; the publisher is typically a good choice.

The Unified CM server is omitted from other CUBE deployment model diagrams below, since it plays no part in call handling.

This basic deployment is unlikely to ever be used in a production environment. More typically a Unified Communications Manager or 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 one at their core. From the strict perspective of CUBE and Cisco MediaSense, all the other models are no different from this one.

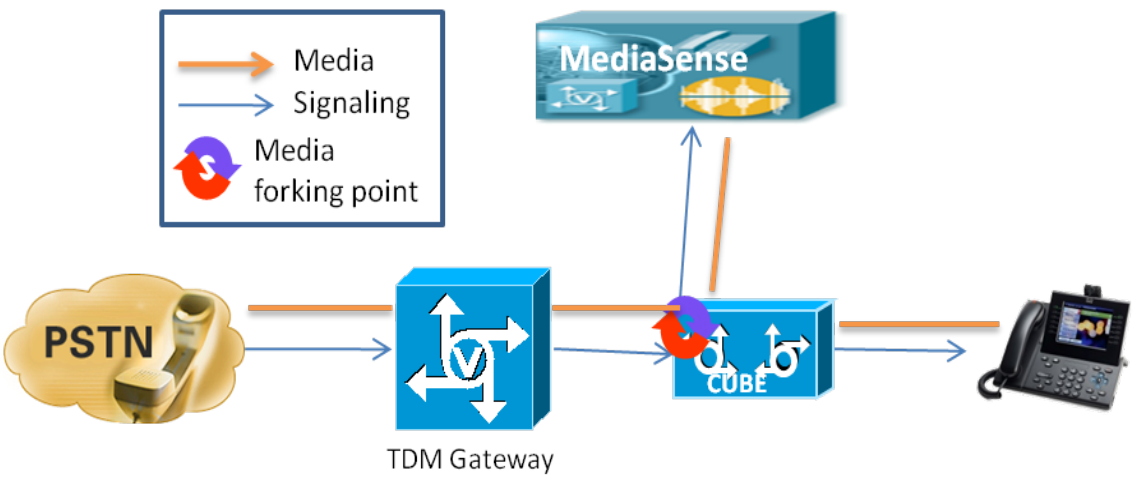
II (a/b/c). Basic CUBE Deployment with Various PBXs



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

The above diagram shows three ways in which Cisco 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's experience will be recorded. Additional combination are possible as well; for example a call may be handled by an IP-based or TDM-based IVR system.

II (d). 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 via TDM, then a separate TDM gateway should be provisioned as shown in the diagram above. Forking should 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.

III. CUBE Deployments with CVP

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

- III (a). CUBE Deployments with Unified CVP – Centralized, SIP Trunks, No Survivability
- III (b). CUBE Deployments with Unified CVP – Centralized, SIP Trunks, With Survivability
- III (c). CUBE Deployments with Unified CVP – Centralized, TDM Trunks
- III (d). CUBE Deployments with Unified CVP – Centralized, Outbound Dialer

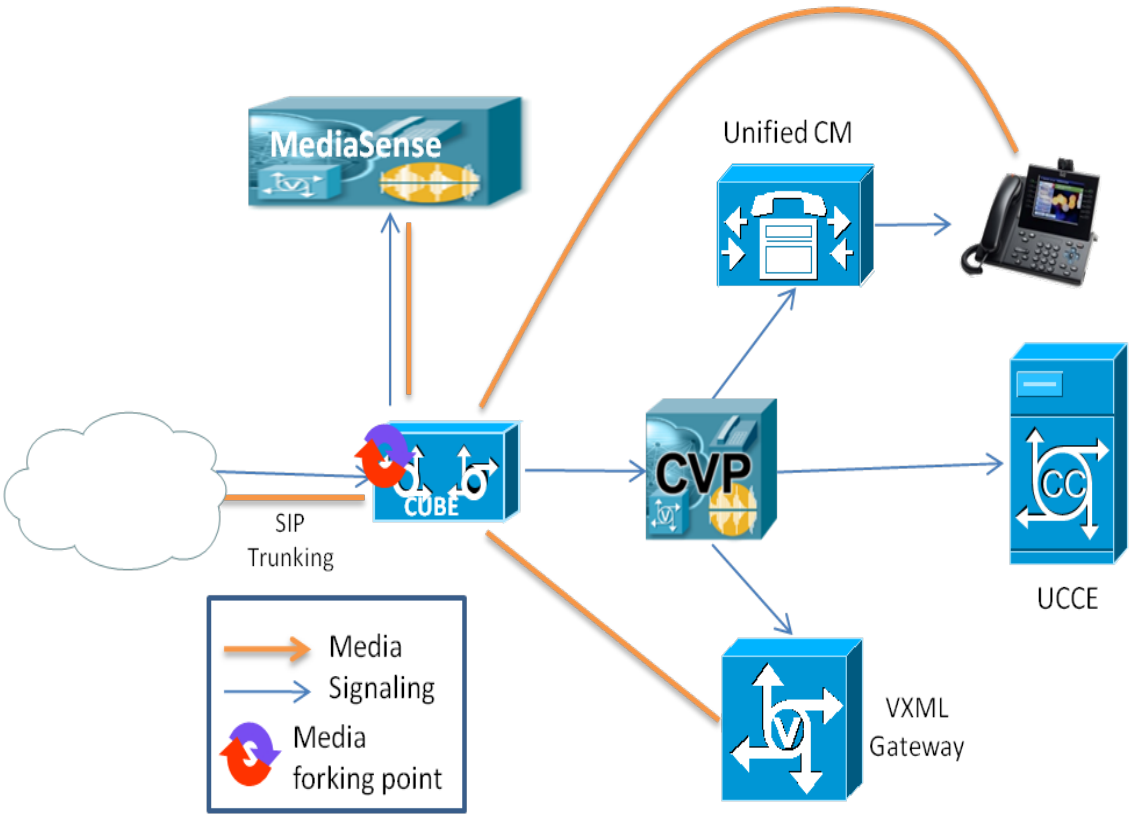
CVP deployments typically involve a VXML function and optionally a TDM to IP conversion function. CVP deployment recommendations sometimes provide for combining those two functions on the same ISR router. There are also CVP deployments which involve incoming SIP Trunks rather than TDM lines. These deployments may actually use CUBE routers, and they too may also host the VXML function.

Technically that same router could be configured to provide media forking capabilities as well, but there are a number of caveats and restrictions. Cisco currently does not recommend doing so. Media forking should be hosted on a separate CUBE device which is specifically provisioned for that purpose.

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 Cisco MediaSense, nor is the content of interest to Cisco MediaSense. The recording dial peer in CUBE must therefore be configured to prevent this content from being forwarded to Cisco MediaSense, by adding the command "signaling forward none" to the recording dial peer.

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

III (a). CUBE Deployments with Unified CVP – SIP Trunks, No Survivability

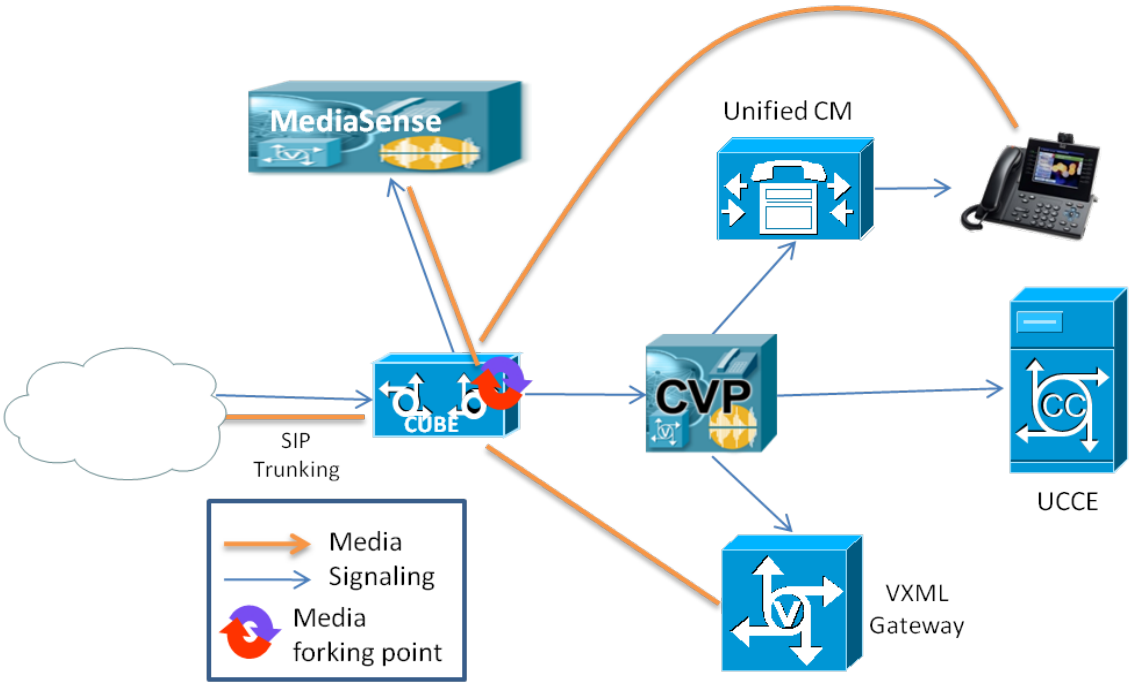


In this scenario, Unified CVP should manage all call control operations, including (typically) an initial delivery to a VXML gateway for music on hold or other treatment, a subsequent delivery to a Cisco Unified Contact Center Enterprise (Unified CCE) agent, and possible further network transfers to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

As long as Unified CVP is configured as recommended in its documentation, Unified CVP will affect these transfers by issuing SIP invites 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 should be configured on the outside dial peer.

III (b). CUBE Deployments with Unified CVP – SIP Trunks, With Survivability



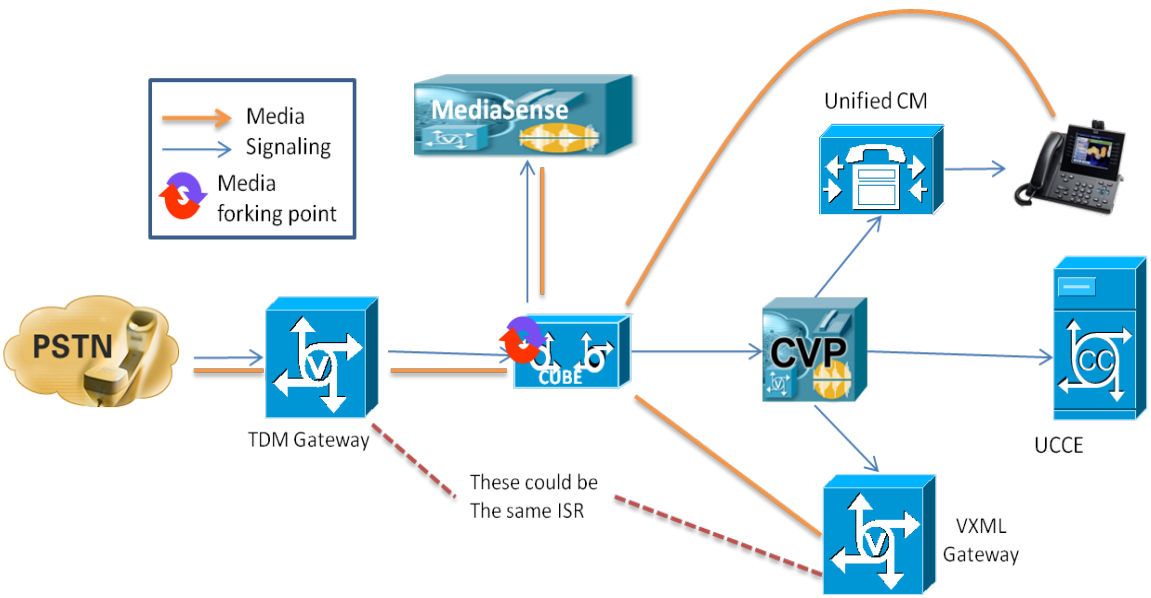
This scenario is identical to the one above, except that the customer has elected to use Unified CVP's Survivability script to manage call failures, time of day routing, etc. To use Unified CVP's Survivability script, you must place it on the outside dial peer in CUBE. IOS does not allow a script and media forking to occur on the same dial peer however, so you must use the *inside* dial peer for media forking as shown in the diagram. As explained above under "II. Basic CUBE Deployment", configuring recording on the inside dial peer is risky because of the possibility that call manipulations may inadvertently trigger IOS to start a new dial peer matching operation. This would terminate the current recording session.

That said, as long as Unified CVP is configured as recommended in its documentation, Unified CVP will affect these transfers by issuing SIP invites to the destination device, rather than to CUBE. This effectively prevents CUBE from triggering a new dial peer match.

Note that if Survivability kicks in to handle a mid-call failure of some kind, any audio played by that script, such as a "technical difficulties" message, cannot be recorded by Cisco 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.

See also additional information about REFER transfers, under "Additional Options and Considerations" below.

III (c). CUBE Deployments with Unified CVP – TDM Trunks



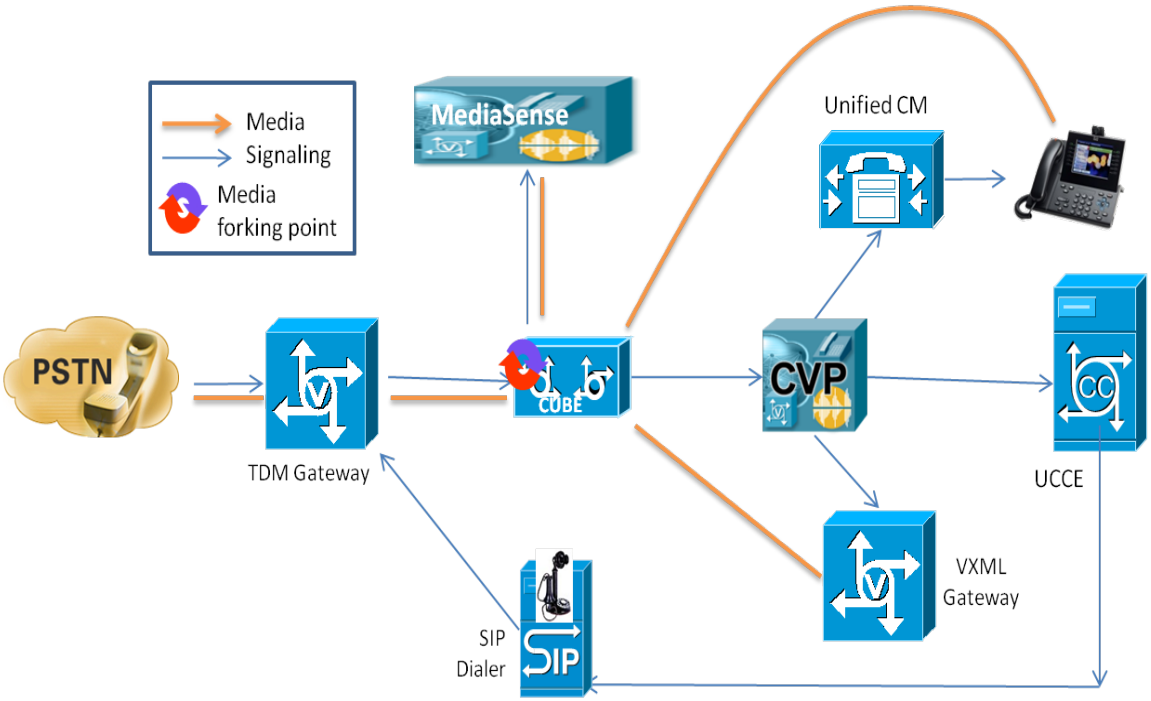
A TDM Cisco MediaSense CUBE deployment for CVP is just like a SIP Trunk deployment, except that a separate TDM gateway is placed ahead of the CUBE. CUBE still does the media forking on the outside dial peer, and CUBE still acts as the router that Unified CVP interacts

with.

If Survivability is used, it should be placed on the POTS dial peer in the TDM gateway, and not in the CUBE. This allows us to keep the media forking on the outside dial peer in CUBE, even though Survivability is enabled.

As mentioned earlier, if Unified CVP will be issuing DTMF tones to the PSTN, as in "*8 Transfer Connect" transfers, be sure to 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.

III (d). CUBE Deployments with Unified CVP – Outbound Dialer



Outbound campaigns using the Unified CCE SIP Outbound Dialer should be 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 should instruct the TDM gateway to connect the call via CUBE to Unified CVP. From the perspective of CUBE and Cisco MediaSense, this appears to be just another inbound call.

Note that the outbound dialer must be connected to the TDM gateway, not to the CUBE.

Additional Deployment Options and Considerations

Redundant Media Forking using CUBE

Earlier, we stated that the best practice is to 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 will result in two independent recording sessions. The dial peers may even be configured to deliver recordings to two separate and independent Cisco MediaSense clusters, implementing true recording redundancy. However, doing so will severely impact 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 can be 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 discussion applies to contact center recording where Unified CVP is used for call routing.

By forking media from CUBE, you are able to record the entirety of the caller's experience. This includes not only his conversation with one or more agents, but also any VRU or call queuing activity which 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.

But some customers may find it desirable 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.

As Unified CVP is typically configured, when an agent becomes available, Unified CVP sends a SIP Invite to the Unified CM which controls that agent's phone. Unified CM negotiates with the ingress CUBE to connect the media stream from the CUBE to the agent's phone. CUBE itself never gets involved in routing that segment of the call, since it never needs to figure out what IP address handles the selected agent's extension.

But Unified CVP can also be configured such that the agent-segment Invite gets sent to the CUBE rather than to the Unified CM. The configuration can be done in any way you prefer to configure your routing in Unified CVP. It could be done via Local Static Routes, via an Outbound Proxy Server, or via 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 CUBE which is specifically for routes to agent extensions, with Unified CM as the destination target. Then you configure media forking on those dial peers only. This technique will allow you to record the agent segment of the call.

There are some caveats, however. First, notice that this arrangement once again causes the call to flow through CUBE a second time. As a result, the call carrying capacity of CUBE is reduced by the percentage of calls that are connected to agents. Second, this technique has the potential to confuse Unified CM's Call Admission Control (CAC) algorithm. When CVP transfers the call to the agent, it specifies a DN which Unified CM believes will create a direct media connection between the gateway and the agent's phone, when in fact it is creating a media connection that passes through the CUBE first. If the CUBE is colocated with either the phone or the gateway, then this is not a problem, since the actual WAN traversal is not affected. But if the CUBE is located at some third site, then bandwidth will be used for which Unified CM is not able to account. An example of such a scenario is a two data-center deployment in which calls arriving in either data center get routed to a shared set of agents in a third location. A call might arrive at data center A, and then be routed through a CUBE in data center B before being delivered to the call center at site C. CVP has no way to control which CUBE the call ultimately gets routed through, so it cannot force the call to pass through a CUBE which is located in data center A. As a result, *this technique to avoid recording the VRU segment of the call is not recommended in multi-site deployments.*

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 will no longer be 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 will result in CUBE itself executing the transfer, taking Unified CVP out of the loop, but keeping CUBE itself in the signaling and media path. CUBE can thereby continue to record the call. You can accomplish this by adding "voice service voip; no supplementary-service sip refer" to your CUBE configuration.

Also, note that 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 obviously 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.

Media Forking on a TDM gateway

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.

This is not a recommended call flow for production deployments. 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 will now need two routers. It is better to configure the two routers for their individual purposes, rather than to use both routers for both purposes.

Running VXML on a CUBE

In Unified CVP deployments without Cisco MediaSense, it is possible to run VXML voice browser functions on the same router as CUBE. However, Cisco does not recommend sharing media forking and VXML activities on the same router.

Configuration Requirements for Other Solution Components

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

Unified CM

Unified CM must be configured appropriately to direct recordings to the Cisco MediaSense recording servers. This includes configuring a Recording Profile, as well as various SIP parameters. Phone zones must be configured to avoid use of iLBC or iSAC codecs on phones whose media will be recorded. Note that for Cisco MediaSense, SIP over UDP is not supported.

Also, since Cisco MediaSense uses AXL to authenticate users, Unified CM's AXL service must be enabled on at least one of its servers.

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 Cisco MediaSense CUBE deployment requires an AXL connection to a Unified CM for authentication purposes, even if it will not be processing calls. It can be a Unified CM which is already installed and in use for other purposes, or it can be one which is installed specifically for use with Cisco MediaSense. The administrator will need to configure one or more Unified CM End Users and import them into Cisco MediaSense to be used as Cisco MediaSense API users.

Streaming Media Players

Cisco MediaSense has been tested with the following off-the-shelf media players:

- VLC version 1.1.11
- QuickTime
- RealPlayer

Each of these media players has its own quirks. VLC for example, can only play one media track at a time. Also, be aware that these media players are not designed to handle silence. Playback of recordings which include silent segments may produce unpredictable behavior.

Note that none of these players support g.729 or g.722. A custom media player is required in order to play media which was recorded using those codecs. The built-in Cisco MediaSense media player, accessible through the Search and Play application, can play g.729 if running on a Mac machine, and g.722 if running on a Windows machine.

SIP Proxy Servers

SIP Proxy Servers are currently not supported between Cisco MediaSense and Unified CM or CUBE.

Cisco Unified Session Manager Edition

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

Contact Center Environments

- Cisco MediaSense does not explicitly interact with or support Unified Contact Center Enterprise (Unified CCE) or Unified Contact Center Express (Unified CCX). The recording functions which are available with these products' Agent/Supervisor Desktop clients utilize different mechanisms for initiating and capturing recordings and require their own established recording solutions.
- For supervisor whisper feature, Cisco MediaSense does not record the whisper call between agent and supervisor. This is because the agent phone's build-in-bridge 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's* phone is configured for forking, then the whisper will be included in the supervisor's phone recording.
- Equipment which monitors agent conversations by listening to a span port output and filtering on the agent's phone MAC or IP address may not function properly while the phone is also forking media for recording. This is because every RTP packet will be emitted from the phone twice, and the listening device may not exclude those packets which are destined for the recording server from its capture. The result will be an apparent echo observed by the listener. Please discuss this with your silent monitoring vendor if the application calls for monitoring of conversations which are also being recorded.

High Availability

Cisco MediaSense implements a redundant, highly available 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

As mentioned elsewhere, a Cisco MediaSense cluster may contain up to five servers, each capable of recording up to a specific number of simultaneous calls. The methodology differs slightly for Unified CM and CUBE calls, but conceptually there are two phases involved. First, the call controller (Unified CM or CUBE) selects a Cisco MediaSense server (the *pilot* server) to send the initial Invite to, and second, the pilot server redirects the Invite to a potentially more appropriate 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 Invite. The second phase allows Cisco MediaSense servers to balance the load among themselves without any active support from the call controller. The algorithm used is aware of the state of all recording servers within the cluster and will 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 CM should be configured such that it sends its Invites to each server in succession, in round robin fashion. This ensures that recording servers are fully equal to each other in terms of initial SIP Invite preference, and avoids situations in which one server receives the bulk of Invites. As CUBE does not support a round robin distribution, it should instead be configured to always deliver its Invites to one particular Cisco 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, only 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 Invite. The usual SIP processing for both Unified CM and CUBE in this case is to deliver the Invite to the next server in the list, thereby also implementing a form of redundancy. However, it must wait for a timeout to expire before determining that it must try another server. The SIP specification actually envisions it trying the same server several times, with progressively growing timeouts, before determining that the targeted server is unavailable. Since Unified CM and CUBE only involve recording servers *after* the primary media path has already been established, such operations can clearly take much too long for the resulting recording to be useful. Unified CM in fact sets a time limit beyond which, if the recording hasn't begun, it will stop trying. The net result is that if Unified CM selects a recording server which is not responding, the call in question will most likely not be recorded. CUBE does not have such a time limit; such calls will end up being recorded, but a substantial initial segment of the call will be clipped.

To reduce the likelihood of losing recordings due to a recording server failure, Cisco MediaSense works with Unified CM and CUBE to support a facility known as "SIP Options Ping". This allows the call controller to periodically probe each recording server to make sure it is up and running, without having to do so while a conversation is literally waiting to be recorded. Once it is aware that a given Cisco MediaSense server is not running, the call controller will skip that server as it traverses its round robin or sequential list of recording servers, thereby distributing the incoming load across the remaining servers. In single-node deployments however, SIP Options Ping is not recommended. Not only is it not helpful, but it can in fact result in unnecessary failure recovery delays. The Cisco MediaSense User Guide contains instructions for configuring the SIP Options Ping facility as well as other CUBE and Unified CM 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. Similarly, the amount of storage space available for recording session retention will also be impacted.

Recording Server Redundancy - Recordings in Progress

If a recording server fails, all calls which are currently being captured on that server are changed from an ACTIVE state to an ERROR state, and the contents are discarded. Note that the detection of such failed calls, and therefore the state change, may not occur for some time, on the order of an hour or two.

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

Recording Server Redundancy - Saved Recordings

After a recording is completed, Cisco MediaSense retains that recording on the same server which captured it. If that server goes out of service, then none of its recordings will be available for playback, conversion, or download during that period, even though information about them can still be found in the metadata.

Metadata Database Redundancy

The Primary and Secondary servers (which we will call the "database servers" in this section) each maintain a database for metadata and configuration data. They also each implement the Cisco MediaSense API, including the ability 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 may also 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 does fail, then 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 required. (If the failure lasts for an extended period of time, the system will actually raise an alarm, and disable replication completely and reestablish it when the failed server recovers.)

Depending on the duration of the outage and the amount of churn which occurred during the outage, 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.

During the recovery period, some irregularities and inconsistencies between the two servers may be noticed. It is therefore not advisable to rely on the recovering server for API operations until the catch-up operation has completed, a state which can be determined manually via CLI commands.

Eventing Redundancy

An event is generated by one database server as a result of an action which took place on that server. It is not generated by both database servers. For example, if a recording server begins a recording, it initiates a session record in *one* of the database servers. Though 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 therefore subscribe to *both* database servers in order to be sure it receives all events (the two subscriptions may designate the same target URI, however, which does simplify things somewhat).

As a convenience, Cisco MediaSense provides the ability for each database server to itself subscribe to events which are generated by the other, and forward them to subscribers almost as if they had been generated locally (a flag is included in the event body which identifies such forwarded events). This capability can be enabled in the Cisco MediaSense Administration facility. If enabled, a client need only subscribe to one database server or the other. That said, 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. The risk here 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 an event 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 immediate API queries should check the event forwarding flag to ensure that they are addressing their queries to the database server on which the update was generated.

Uploaded Video Playback Redundancy

The load balancing and redundancy mechanism which is used to distribute incoming recording calls is very similar to that used to distribute incoming video playback calls. The call arrives at a *pilot* node, and then is immediately redirected to a *home* node which may be more suited to handling the operation. Cisco MediaSense is able to play an uploaded video as long as at least one of its nodes is able to play it.

Typically, all nodes in a cluster are equally able to handle such a request, because uploaded videos are always 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. An incoming media playback call therefore gets automatically redirected to a node which 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 that of requests for recording audio. Like an audio recording attempt, the person interested in playing a video while in queue or on hold is not at liberty to simply try again later if the system is busy. Therefore, Cisco MediaSense treats incoming video playback requests together with recording requests at a higher priority than RTSP play and monitoring requests. In other words, RTSP play and monitoring requests are subjected to a lower capacity throttling threshold than recording and video playback requests, making it possible for a given node to still accept the latter even while the former are being redirected to other nodes.

Unified CM Failure while Playing Uploaded Videos

If Unified CM fails while Cisco 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 Cisco MediaSense to be informed when the endpoint device hangs up. Therefore, the video simply continues to play until it comes to an end, at which time it terminates and all resources are released cleanly. However, in the case of videos configured to playback repetitively (see "Incoming Call Handling Rules"), the playback may never terminate. For those cases, Unified CM sends Cisco MediaSense a session keepalive message twice every 30 minutes by default. If Cisco MediaSense does not receive two of these timers in succession, it assumes that Unified CM has crashed, and it terminates the playback, releasing resources cleanly.

As a result, repetitive video playbacks can remain active for up to 30 minutes following a Unified CM crash. While they are in progress, Cisco MediaSense considers those media streaming resources to be in use for the purposes of determining whether new incoming calls and streaming requests can be accepted.

The 30 minute figure is a Unified CM Service Parameter which can be modified if desired.

Backup and Restore

Cisco MediaSense permits the use of VM backup mechanisms for the system, metadata, and media disks. The product does not provide its own built-in backup and restore capability. Also note that VMWare's virtual machine snapshot capability should not be employed with Cisco MediaSense except as part of the software upgrade process, due to unpredictable performance impact.

There is an important caveat with respect to VM backups. While Cisco MediaSense functions in a cluster model, VMs are only backed up on a node-by-node basis. When you backup an Expansion node, for example, you are not capturing any of the metadata for recordings on that node, since the metadata is stored (redundantly) on the Primary and Secondary nodes. Similarly, if you backup the Primary node, you are not capturing any actual recordings except those which physically reside on that Primary node.

In a normal isolated-node backup/restore scenario, if you ever need to restore the node from backup, you will lose all information captured since the last backup of that node. With Cisco MediaSense, you will lose all of the *recordings* captured on that node since the last backup of that node, but not their metadata. The redundantly-stored metadata, even if it is the Primary or Secondary node itself which is being restored from backup, will remain intact. Any attempt to play or download the lost media files however, will obviously fail.

The above discussion reflects on system backup to guard against a catastrophic hardware failure. If your goal instead is to selectively preserve individual media files, then those files can be individually converted to .mp4 and downloaded to a separate (customer-provided) server for backup.

Network Redundancy and NAT

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

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

Security

User Administration and Authentication

Cisco MediaSense supports three types of users: API users, application administrators, and platform administrators. There is only one application administrator and one platform administrator, and these are both configured during installation and their credentials are stored within Cisco MediaSense itself.

Any number of API users can be configured, after the installation process has been completed. For these users, Cisco MediaSense makes use of Unified CM's user administration. Any users configured as End Users in Unified CM may be selectively enabled as Cisco MediaSense API users, and once signed in any such user can access all API functions. API clients sign in using a Cisco MediaSense API request, but Cisco MediaSense delegates the actual authentication to Unified CM via the AXL service. API user passwords are maintained in Unified CM only and are not copied to Cisco MediaSense.

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

Cisco MediaSense API and Events

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

Internal Intracluster Communication

For their own purposes, various components in a Cisco 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 nevertheless 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 of course secure by definition, but their security extends only to the transport mechanism. The URIs themselves can be transmitted insecurely by people and equipment which are out of Cisco MediaSense's control.

In order to prevent unauthorized users from making use of these media output URIs inappropriately, Cisco 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 will be given access to the recorded media. This authentication will usually be very fast, but it may occasionally take up to 4 seconds to complete.

Uploaded Media Files

Administrator credentials are required in order to upload videos for ViQ, VoH and VoD purposes. The administration screens include a link which can be used to *download* a previously uploaded MP4 file. Though administrator credentials are initially required in order to obtain this link, the link does not itself require credentials, and therefore cannot be considered to be 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 the disk throughput requirements are met. Provisioning and configuring SAN encryption is outside the scope of the Cisco MediaSense product.

Reporting and Call Correlation

The information available in Cisco MediaSense's metadata database is limited to a) that which is provided in CUBE's or Unified CM's SIP Invite; b) tags which are inserted by client applications; and c) information which is generated within Cisco MediaSense itself. Real time correlation with other components is best accomplished using an identifier known as the Cisco-Guid. This identifier is usually created by an IOS device such as CUBE or a gateway, or by Cisco's Customer Voice Portal (CVP) product, and forwarded to other components that the call encounters. It can therefore be used to correlate calls across components both in real time and historically.

That said, for Unified CM calls, Cisco MediaSense does not receive the Cisco-Guid. There are other identifiers which can be used to correlate recordings in Cisco MediaSense with historical call records in other solution components, but the only way to do so deterministically in real time is to have a TAPI or JTAPI connection with the Unified CM. The association can also be achieved using the device extension, which is available in both Cisco MediaSense and Unified CCE. However, the correlation is somewhat non-deterministic in complex environments. This is because in Unified CM, *lines* are configured for recording, not *devices* or *extensions*, and there is not necessarily a one-to-one correspondence. 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.

A detailed discussion of call correlation techniques can be found online at http://docwiki.cisco.com/wiki/FAQs_for_Cisco_MediaSense .

Serviceability and Administration

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

It also provides a number of entry points to support system *serviceability* functions, covering all the functions necessary to service the product. Cisco MediaSense offers most of these functions through the Real Time Monitoring Tool (RTMT), which is similar to Unified CM and other Cisco voice products. RTMT is a thick client which can be downloaded onto any Microsoft Windows system (other operating systems are not supported) from the Cisco MediaSense serviceability web pages. It provides the following capabilities:

- Collecting log files of specific types and specific time periods from some or all Cisco MediaSense servers. Remote log browsing is also available so that logs can be viewed without having to download them
- Displaying Alerts, including the current set of System Conditions. System Conditions are service impacting conditions, such as 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, for example, the amount of space in use on the media partitions, or the number of recordings in progress at any given time. RTMT also allows thresholds to be configured for these values; crossing such a threshold creates an entry on the Alerts screen. As with System Conditions, these alerts may also be sent to a SYSLOG server, or trigger proactive notifications to be sent by email.

Separate from RTMT, Cisco MediaSense provides a specialized browser-based Serviceability User Interface which provide administrators with the following capabilities:

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

Finally, Cisco MediaSense supports a Command Line Interface (CLI) for many additional service functions. Administrators of the Unified CM will already be familiar with most of these functions.

Note that SNMP is not supported at this time.

Best Practices

This section describes some best practices for consideration when preparing a Cisco MediaSense deployment.

Proactive Storage Management

As mentioned elsewhere in this document, Cisco 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. Cisco MediaSense can be configured to automatically clean up these elements as well; otherwise clients must still take responsibility for managing disk space proactively.

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

If you are using Retention Priority, then there will be 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 should be invoked using the Cisco MediaSense API; details can be found in the Cisco MediaSense Developer Guide. If these activities are going to be performed regularly, it is advisable to schedule them for low usage periods in order to minimize possible impact on normal operations.

Media Storage Space Provisioning

At the application level, Cisco 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 however, and beneath these, at the VM level, each one is made up of 1 to 16 virtual disks. These virtual disks are then in turn mapped via VMWare host configuration to physical disks in the server or on a SAN. And finally, the physical disks are configured and managed using a RAID controller.

These physical disks must meet certain speed and throughput requirements, as described in the "Storage" section in the Compatibility Matrix below. In particular, all *non-media* disks must be configured as RAID-10, and ESXi "thin provisioning" is not supported on any disk.

SIP Configuration

The following best practices apply to CUBE deployments

- In CUBE deployments, use SIP early offer on dial peers which go to Cisco MediaSense. This is the default setting. Unified CM only implements delayed offer, and does not provide an option.
- Use SIP over TCP on dial peers or Trunks which go to Cisco MediaSense.
- Configure SIP Options Ping support for dial peers or Trunks which go to Cisco MediaSense (except in single-node deployments). This feature greatly improves failover support for multi-server Cisco MediaSense deployments.

Codec Configuration for Phones

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

Network Provisioning

CUBE Interfaces which carry RTP media MUST be configured to a fixed 1 gigabit speed or higher, and full duplex. Do not rely on auto-negotiation. Sometimes auto-negotiation fails when 100 megabit speeds are available, and even if 100 megabit speeds are properly negotiated, they are not fast enough to handle a heavy call load.

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

Using Scalable Queries

Cisco 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 Cisco MediaSense servers, it is possible to formulate queries that have a significant impact. Cisco 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 should therefore read and adhere to the guidelines for writing scalable queries, which can be found in the Cisco MediaSense Developer Guide.

Distribute HTTP Download Requests Over Time

Some customers will use the HTTP Download facility to create copies of all recordings, using Cisco MediaSense more as a temporary location for these files than as a long term archive. This is a perfectly valid use case. However, customers may be tempted to batch up such download 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. One may, for example, use the session ENDED event to trigger a download as soon as the call recording terminates.

Alarm Monitoring

Various situations which require administrator attention cause alarms to be raised in the form of System Conditions. These can be observed in the system logs, as well as in RTMT's alarms page. Using RTMT, it is also possible to configure these alarms to be sent to a SYSLOG server, and to send email messages to a designated email address. Cisco MediaSense does not currently support SNMP alarms.

Cisco recommends that one or more of these methods be used to actively monitor the state of the Cisco MediaSense servers.

Compatibility Matrix

This section lists specific versions, model numbers and specifications of components with which Cisco MediaSense is compatible.

Server Platforms

Cisco 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 will provide, however; details can be found at http://docwiki.cisco.com/wiki/Specification-Based_Hardware_Support.

For Cisco MedaSense, a detailed list of TRC models and supported server specifications can be found at http://docwiki.cisco.com/w/index.php?title=Unified_Communications_Virtualization_Supported_Applications. Note that other than the TRC models (which are manufactured by Cisco), only Hewlett Packard (HP) and IBM servers are supported, subject to the stated minimum performance specifications.

Server configurations can be divided into those which have Direct Attached Disks (DAS) and those which do not. For diskless servers, it is a requirement to provision fiberchannel SAN. For DAS servers, fiberchannel SAN is an option. In either case, 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 which are laid out in this document. Non-media disks must be configured as RAID-10, which is *not* their off-the-shelf default configuration. Configuring them as RAID-10 will necessarily reduce the amount of available storage space.

The only server not listed in the docwiki pages referenced above is the SRE module which runs inside a Cisco ISRG2 router. The ordering code for that device is **SM-SRE-910-K9**. Be sure to order 8GB memory.

When ordering C-series servers, be sure to include battery backup for the write cache.

Hypervisor

ESXi Version	Notes
ESXi v4.0	Enterprise Plus edition
ESXi v4.1	Enterprise Plus edition With LRO disabled
ESXi 5.0	**Standard edition** With LRO disabled
SRE-V 2.0	Available from Cisco at http://developer.cisco.com/web/srev With LRO disabled

Note that a hypervisor is required. Cisco MediaSense is not designed to run on bare metal hardware.

Storage

Cisco 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. Note that *thin provisioning* is not supported for any Cisco MediaSense disks.

Recorded Media Storage. Up to 60 terabytes is supported per cluster, divided into 12TB in each of five servers. SRE modules support a fixed 210GB of media storage per node.

Uploaded Media Storage. Uploaded media requires nowhere near that amount of storage, but we state here for completeness that it can also support up to 60 terabytes, divided into 12TB in each of five servers. Cisco does not support the media upload and playback use cases on SRE modules.

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 on Cisco's list of supported SAN products (see "Cisco Unified Communications on the Cisco Unified Computing System" at http://www.cisco.com/go/swonly, under the heading "Required Storage"). Also, SAN storage must be engineered to meet or exceed the following performance specifications for each MediaSense Virtual Machine. These specifications below 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 below specifications can still be met.

Average IOPS for media partitions:	50 ops/sec
Average bandwidth for media partitions:	1000 kbytes/sec

Database Storage. The *second* disk on Primary and Secondary servers maintain the metadata database. These disks are very active, and have the the most stringent requirements. Most installations will use direct attached disks for this purpose. Such disks *must* be configured as RAID-10 in order to maximize throughput.

If you are using diskless equipment, then you must ensure at least the following performance specifications:

Average IOPS for database partitions:	1700 ops/sec
Average bandwidth for database partitions:	7300 kbytes/sec

Operating Software Storage. The *first* disk on all server types, as well as the *second* disk on Expansion servers, have much lower performance requirements than the database disks. Most installations will use direct attached disks for this purpose. Such disks should be configured as RAID-10 to be consistent with the database disks.

If you are using diskless equipment, then you must ensure at least the following performance specifications:

Average IOPS for O/S partitions:	100 ops/sec
Average bandwidth for O/S partitions:	2000 kbytes/sec

SRE Modules. Each SRE module comes equipped with two 500GB drives. These must be manually configured as RAID-1, making a total of 500GB available to the hypervisor. The required downloadable .OVA template automatically carves this disk into two 80GB drives and one 210GB drive, formatted. This is the only configuration which is allowed on SRE installations.

Other Solution Component Versions

Cisco MediaSense depends only on the versions of Unified CM and CUBE. There is no dependency on particular versions of Unified CVP or Unified CCE. However, for deployments which include both Cisco MediaSense and Unified CVP, where the two products will be sharing the same router, the IOS version running on that router must be one which is compatible with both products. At the time of this writing, IOS versions 15.2(2)T4 and 15.2(3)T meet that requirement, but it is important to verify it with both products' compatibility matrices at deployment time.

Component	Releases Supported
Unified CM	9.1, 9.0, 8.6, or 8.5
CUBE IOS	15.2(2)T4 or later rebuild (includes fix for CSCuc87726) 15.2(3)T or later rebuild 15.3(1)T or later rebuild

Phones

- For CUBE-based forking, all Cisco phones are supported. In the case of video calls however, only the audio portion of those calls can be recorded.
- For endpoint-based forking, all Cisco phones which support media forking are supported. Note that none of the Cisco phones are currently capable of forking video.
- For direct recording, all Cisco phones are supported, both for audio and video media.
- For playback of uploaded videos, only the Cisco EX-60 and EX-90 telepresence endpoints are supported.

Following is a partial list of supported devices.

Endpoint Category	Endpoint Forking	CUBE Forking	Direct Recording	Models
Audio Hard Phones	- Audio	- Audio	- Audio	7906, 7911, 7921, 7925, 7941, 7942, 7945, 7961, 7962, 7965, 7970, 7971, 7975 An up to date list may be found under "Unified CM Silent Monitoring and Recording Device Support" at http://developer.cisco.com/web/sip/wikidocs
Soft Phones	- Audio	- Audio - Audio streams from video calls	- Audio - Video	Cisco IP Communicator (CIPC) v7.0(1) or later (with video when paired with Cisco Unified Video Advantage (CUVA))
Video-Capable Phones	- Audio	- Audio - Audio streams from video calls	- Audio - Video	9971, 9951 and 7985, plus any audio phone when paired with Cisco Unified Video Advantage (CUVA). A complete list of these may be found at http://www.cisco.com/go/cuva.
Telepresence Endpoints	Not Supported	- Audio - Audio streams from video calls	- Audio - Video	EX-60, EX-90 Audio must be configured for g.711 or g.722 codec

Web Browsers

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

Browser	Host Operating System
Internet Explorer 9	Windows
Firefox 18	Windows

Other browsers and versions may be used and will likely function correctly. However, any erroneous behavior discovered will be addressed by Cisco only on a best-effort basis, if the problem cannot be reproduced in one of the above browser versions.

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 11, 64-bit
Microsoft Windows	JDK or JRE 7 update 11, 32-bit

Cisco MediaSense Upgrades

Cisco MediaSense Release 9.1(1) software may be installed as an upgrade from Release 9.0(1). Systems running Release 8.5(4) must first be upgraded to 9.0(1). Upgrades from prior releases are not supported.

There have been no changes in the Cisco MediaSense API between Releases 9.0(1) and 9.1(1). Beyond that, all of the Cisco MediaSense APIs are upward compatible with those of Release 8.5(4), but client software may need to be modified in order to provide HTTP-BASIC

credentials, and to handle a 302 redirect. HTTP-BASIC credentials must now be provided with all RTSP and HTTP download requests.

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

There is a new feature in Release 9.1(1) which permits recorded media storage to be increased in size after installation. This feature is not available in upgraded systems, however. It only functions in systems that have been fresh-installed with Release 9.1(1). The new uploaded media partition on the other hand, also introduced in Release 9.1(1), is automatically created during upgrade, and *does* support the ability to be increased in size after installation.

If you upgrade a Cisco MediaSense cluster from 9.0(1) to 9.1(1), and then wish to add nodes to your cluster, please be aware that, though the new nodes will be installed with expandable recorded media storage, Cisco still recommends that you do not take advantage of that flexibility. The recommendation is to provision approximately the same amount of recording space on the each *new* node as was available on each *upgraded* node. Though storage space disparity across nodes in the cluster does not present a problem for the Cisco MediaSense product, it could result in pruning ahead of the configured retention period on smaller nodes, but not on the larger nodes. Administrators may find this behavior confusing and unpredictable.

Scalability and Sizing

Performance

Each rack-mounted Cisco 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. 400 is a total of all streams used for recording, live monitoring, playback, MP4 conversion, and HTTP download, all of which may occur in any combination. The latter two activities are of course, not strictly speaking streaming activities, but they do use system resources in a similar way and are considered to have equal weight. Playback of video track takes about 9 times the resources 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.

Each SRE-based Cisco MediaSense server supports up to 60 media streams (30 calls), at an average arrival rate of 20 calls per minute, with a fixed recording disk space of 200 gigabytes.

In determining how many streams are in use at any given time, you will need to predict the number of onsets for each activity per unit time, as well as their durations. Recording, live monitoring and playback of course have a duration which 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 such video callers typically remain on Hold. MP4 conversions and HTTP download durations can be safely estimated to be about 5 seconds per minute of recording.

Essentially, to determine the number of servers required, one would simply evaluate:

- the number simultaneous audio streams needed, plus 10 times the number of videos being played, divided by 400 (for rack-mount servers), or 60 (for SRE modules);
- the number of busy hour call arrivals per second divided by two; and
- the space required for retained recording sessions divided by 12 terabytes (for rack-mount servers) or 210 gigabytes (for SRE modules)

The number of servers required is equal to whichever of the above is greater, rounded up.

Another factor which significantly impacts performance is the number of Cisco MediaSense API requests in progress. This is limited to 15 at a time (3 at a time for SRE modules), with the ability to queue up to 10 more (or 3 more for SRE modules). These numbers are per node, but they can be doubled for Cisco MediaSense clusters which contain both a Primary and a Secondary node.

Cisco MediaSense does not enforce these limits, with the exception of the number of concurrent API requests. It does however enforce a higher set of limits in order to protect itself from overload conditions which could affect its ability to perform vital functions. See "System Resiliency and Overload Throttling" above for more information.

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

Hardware Profiles

When Cisco MediaSense nodes start up, they adjust their capacity expectations according to the hardware resources they discover from the underlying virtual machine. The following table indicates the relationship between virtual machine resources and supported capacities. The table presents information on a per-node basis. When the server is installed using one of the Cisco-provided OVA templates, the correct amount of CPU and memory are automatically provisioned, and either the "Standard" or "SRE" profile will be selected from the table below. 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 rows below, the server is considered to be "Unsupported", and the capacities in the "Unsupported" category are used.

Profile Name	vCPUs	Memory	Audio-Weight Media Streams Supported	Concurrent API Requests Supported
Standard	**7**	**16 GB**	400	15 + 10 queued
SRE	**2**	**6 GB**	60	3 + 3 queued
Unsupported	**Anything else**	**Anything else**	a few	1 + 1 queued

Cisco MediaSense also adjusts a number of other internal parameters based on the profile selected.

Maximum Session Duration

Cisco MediaSense can record calls which 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 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 such calls, the call arrival rate, duration and duty cycle, and the retention period desired. Since most of these parameters are very difficult to estimate, we will focus here on only the number of recording session hours, and the retention period. Essentially, we will answer this question, "How much disk space do I need in order to retain *h* hours of recordings for *d* days?"

We begin by selecting the codec. We will assume g.711, which requires about 1MB per minute of dual-stream recording. g.722 also requires roughly 1MB per minute of dual-stream recording. g.729 uses a variable rate compression, which means that the space requirement depends on the content, which is not predictable. On the other hand, for estimation purposes it is generally safe to assume that it requires about one eighth the space needed by g.711, or 128 kilobytes per minute of dual-stream recording. H.264 video is even less predictable; not only does it use a variable rate compression, but it also depends on video resolution, screen dimensions, and a number of other factors. This is best evaluated empirically.

Given g.711 then, we have a rate of 1MB per minute, or 60MB per hour. Assuming the maximum direct-attached storage, 4TB of disk space therefore can store about 70,000 hours of dual-stream recordings. If you have 100 phones which are in active use 80% of the time, 24 hours a day, then you are recording at a rate of 80 hours per hour of elapsed time. That will use up 70,000 hours in 875 hours of elapsed time, or a little over 36 days, after which the oldest calls will need to be pruned. Therefore, your retention period will be 36 days.

Let's say all of the same parameters apply, except that your business is only open 12 hours per day. That will give you a retention period of 72 days.

If you only have 50 agents, active 12 hours a day, then your retention period rises to 144 days.

Finally, all of the above assumes 4TB of storage space available. If you deploy five Cisco MediaSense servers, each with its maximum SAN storage allocation, then you have in effect, 60TB available. In that scenario, the retention period for 50 agents at 12 hours per day with an 80% active usage ratio would be 2160 days, or about six years.

Here is the formula:

```
Codec bit rate (B) in MB/hour for two streams
Number of phones (P)
Average Usage ratio of each phone (U) in hours per day

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

Total Storage available across all servers (S) in GB

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

There is one more factor to consider however: conversions into .mp4. 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 anecdotal trials, .mp4 averaged about 18 MB/hour for dual-channel audio, and about 180 MB/hour for audio+video. (Note that the .mp4 files use AAC, which is another 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 obviously reduces the retention period. Thus the Write Rate now becomes:

```
Proportion (M) of recorded session hours which are converted to .mp4 and retained
.mp4 average bit rate (K) in MB/hour

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

Notwithstanding the above calculations, there is also an absolute maximum number of recordings that Cisco MediaSense can retain, no matter how much disk space is provisioned or how long your recordings are. 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. 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 therefore need to deploy multiple ISR G2 routers in order to handle the required call capacity. A single Cisco 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 CM 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 CM will 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 CM administrator should include enough bandwidth for *two 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

For video playback use cases, even though Cisco MediaSense will only be sending data and not receiving it, the endpoint does not know that in advance, and the media connection is therefore negotiated to be bi-directional. This could be an important consideration, since video typically requires a large pipe, and the use of bi-directional media essentially implies that you must provision double the bandwidth that you might have otherwise expected.

Impact on Unified CM Sizing

Cisco MediaSense does not connect to any CTI engines, so the CTI scalability of Unified CM is not impacted. However, when Cisco MediaSense uses Cisco IP Phone built-in-bridge recording, the Unified CM 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, go to http://developer.cisco.com/web/sip/docs, and look for "Cisco Unified CM Silent Monitoring & Recording Overview.ppt" under SIP Trunk documents.