High availability

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

Recording server redundancy - new recordings

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

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

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

Unified Communications Manager is configured so that it sends invitations to each server in succession, in round-robin fashion. This ensures equal distribution of initial SIP invitation preference to all recording servers and avoids situations where one server receives the bulk of the invitations. As CUBE does not support a round-robin distribution, instead it is configured to always deliver invitations to one particular MediaSense server, with a second and perhaps a third server configured as lower preference alternatives. If possible, it is best to target an expansion server rather than a primary or secondary server for the pilot role because expansion servers are typically doing less work at any given time.

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

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

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

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

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

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

Recording server redundancy - recordings in progress

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

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

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

Recording server redundancy - saved recordings

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

Metadata database redundancy

The primary and secondary servers (the 'database servers' in this section) each maintain a database for metadata and configuration data. They also each implement the MediaSense API and include the capability to publish

events to subscribed clients. Once deployed, the two database servers are fully symmetric; the databases are fully replicated such that writing to either one causes the other to be updated as well. Clients address their HTTP API requests to either server and use the alternate server as a fallback in case of failure.

Database server failure

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

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

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

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

Event redundancy

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

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

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

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

Uploaded video playback redundancy

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

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

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

Unified Communications Manager failure while playing uploaded videos

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

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

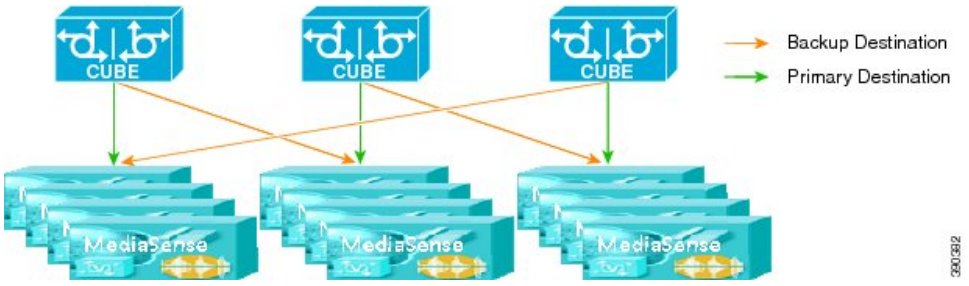
As a result, repetitive video play-backs can remain active for up to 30 minutes following a Unified Communications Manager failure. MediaSense considers those media streaming resources to be in use, or unavailable, for the purposes of determining whether new incoming calls and streaming requests can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

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

MediaSense cluster redundancy

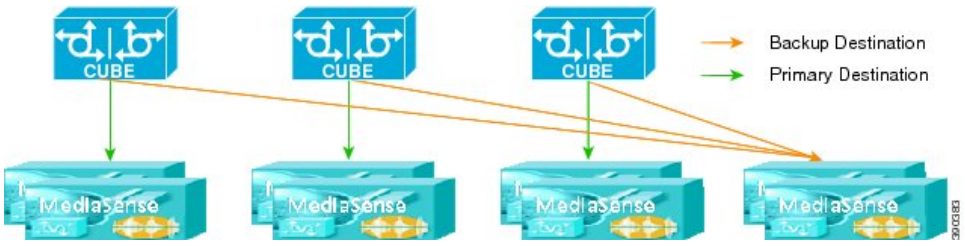
The individual nodes within a MediaSense cluster act in such a way so that if one node fails, the remaining nodes in the cluster automatically take up the slack—assuming they have available capacity. MediaSense clusters can also be configured to take over for one another in the event of an entire cluster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

Ring with spare capacity



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

Spare cluster



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

Configuration methodology

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

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

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

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

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

Backup and restore

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

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

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

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

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

Network redundancy and NAT

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

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