Cisco MediaSense Developer Guide, Release 11.0(1)

First Published: July 02, 2015

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

CONTENTS

Preface

Change History

This document may be updated at any time without notice. Obtain the latest version of this document online at http://www.cisco.com/c/en/us/support/customer-collaboration/mediasense/products-programming-reference-guides-list.html.

Visit this cisco.com website periodically and check for documentation updates by comparing the revision date (on the front title page) of your copy with the revision date of the online document.

The following table lists the change history for this document.

Change	See	Date
Added an API called *getArchiveSessions* to the *Session Query APIs* section.	getArchiveSessions, on page 69	Initial Release of Document for Release 11.0(1)
Added agent information (loginId, lastName, firstName, loginIdDomain, and loginName) to the response schema of the *Session Query APIs* section.	Session Query APIs, on page 63	Initial Release of Document for Release 11.0(1)
Added *lineDisplayName* parameter to the response schema of the*Session Query APIs* section.	Session Query APIs, on page 63	Initial Release of Document for Release 11.0(1)

Change	See	Date
Added parameters *errorDetail* and *errorCode* to the response schema of the*Session Query APIs* section.	Session Query APIs, on page 63	Initial Release of Document for Release 11.0(1)
Updated the Successful JSON Response Schema of the "Session Query" section.	Session Query APIs, on page 63	Initial Release of Document for Release 10.5(1)_SU1
Updated the Event Schema of the "Session Event" section.	sessionEvent, on page 131	Initial Release of Document for Release 10.5(1)_SU1
The following API Parameters are now HTTPS URLs rather than HTTP URLs. • downloadUrl • httpUrl • mp4Url • wavUrl	Shared Parameters, on page 112	Initial Release of Document for Release 10.5(1)
Added the section,*Unified Communications Manager Network-based Recording*.	Cisco Unified Communications Manager Network-based Recording, on page 8	Initial Release of Document for Release 10.5(1)
Added an API called *getAssociatedSessions* in the *Session Query* section.	getAssociatedSessions, on page 69	Initial Release of Document for Release 10.5(1)

Related Documentation

Document or Resource	Link
Cisco MediaSense Documentation Guide	http://www.cisco.com/c/en/us/support/customer-collaboration/mediasense/products-documentation-roadmaps-list.html
Cisco.com site for Cisco MediaSense	http://www.cisco.com/c/en/us/support/customer-collaboration/mediasense/tsd-products-support-series-home.html
Doc Wiki for Cisco MediaSense	http://docwiki.cisco.com/wiki/Category:Cisco_MediaSense
Troubleshooting tips for Cisco MediaSense	http://www.cisco.com/c/en/us/support/customer-collaboration/mediasense/products-troubleshooting-guides-list.html

Document or Resource	Link
Virtualization for Cisco MediaSense	http://docwiki.cisco.com/wiki/Virtualization_for_Cisco_MediaSense
Frequently Asked Questions for Cisco MediaSense	http://docwiki.cisco.com/w/index.php?title=FAQs_for_Cisco_MediaSense

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation*, at: http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation as an RSS feed and delivers content directly to your desktop using a reader application. The RSS feeds are a free service.

Field Alerts and Field Notices

Note that Cisco products may be modified or key processes may be determined important. These are announced through use of the Cisco Field Alert and Cisco Field Notice mechanisms. You can register to receive Field Alerts and Field Notices through the Product Alert Tool on Cisco.com. This tool enables you to create a profile to receive announcements by selecting all products of interest.

Log into www.cisco.com and access the tool at: http://www.cisco.com/cisco/support/notifications.html.

Troubleshooting

A Troubleshooting Tips wiki provides information to help resolve issues already reported by other users, which is available at http://docwiki.cisco.com/wiki/Troubleshooting_Tips_for_Cisco_MediaSense.

Documentation Feedback

To provide comments about this document, send an email message to the following address:

mailto: ccbu_docfeedback@cisco.com

We appreciate your comments.

CHAPTER 1

Introduction

MediaSense Concepts

This section identifies and defines common MediaSense concepts and terms.

Mapping a Session to a Recording

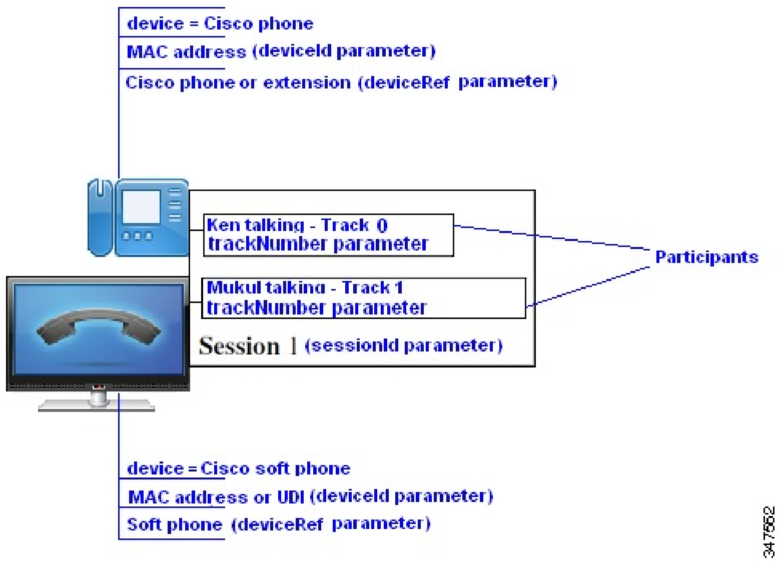
In the context of MediaSense and this document, a session is a recorded monologue, dialog, or conference, which can involve one or more participants. A session in MediaSense has the same meaning as a recording session in Unified Communications Manager. For more information on recording sessions, see the *Cisco Unified Communications Manager Features and Services Guide* available at http://www.cisco.com/en/US/partner/products/sw/voicesw/ps556/prod_maintenance_guides_list.html.

The participants use a device to conduct this session.

A device is a physical entity that can be an endpoint or a personal computer and refers to any item that can be recorded. A device is identified by a deviceRef, which is a phone number or extension for each device.

The deviceId is the unique identifier for each device and it corresponds directly to the name of the device (like the MAC address or Universal Device Identifier [UDI]), as shown in the following graphic.

Figure 1: Device, Device ID, and Device Ref


device = Cisco phone
MAC address (deviceId parameter)
Cisco phone or extension (deviceRef parameter)
Ken talking - Track 0
trackNumber parameter
Mukul talking - Track 1
trackNumber parameter
Participants
Session 1 (sessionId parameter)
device = Cisco soft phone
MAC address or UDI (deviceId parameter)
Soft phone (deviceRef parameter)
347562

Each session is identified by a sessionID and contains one or more tracks. Each track is specific to one audio stream or one video stream, and is identified by a trackNumber.

A session can be live (active) or recorded (completed). You can monitor and record a live session at the same time. You can play back a recorded session at any time. You can use the sessionId to manage recorded sessions.

Several recordings are possible using the Cisco MediaSense platform:

- Forked media from a Cisco IP Phone or Cisco Unified Border Element device. This recording has two audio channels.
- Direct call to/from the MediaSense platform to any phone. This recording has one audio channel and one optional video channel. These recordings are referred to as Blog recordings in this document.

MediaSense recording sessions using the following codec options:

- Audio recordings: g.711 (aLaw or µ-Law), g.722, or g.729 (a or b) codecs.
- Video recordings: h.264 baseline codecs (only with 48 kHz audio sampling rate).

Mapping a Media Stream to a Track

A media stream refers to the packets going through an audio channel or video channel in a live or recorded session.

A track identifies each media stream and quantifies it with additional data such as participants, duration, startDate, and a trackNumber. Track 0 contains media streamed from the forking device, and Track 1 contains media streamed to the forking device. In a sessionEvent, each track is associated with one participant.

The trackNumber assignment is arbitrary and is based on whether the call is incoming or outgoing from the dial peer on which media forking is enabled. For more information, see the section Differences Between Unified Communications Manager and Unified Border Element Scenarios, on page 9.

Hold-Resume or Transfer-Conference Behavior

Unified Border Element sends metadata information that MediaSense uses to associate a Unified Border Element call with the media streams and identify the call participants. The metadata that is accumulated within the bounds of a SIP Session is exposed to clients in the form of a recording session object. A recording session in both Unified Communications Manager and Unified Border Element contains information on tracks, call records, and media events.

When call participants use call features such as hold/resume or transfer/conference, the behavior is different between Unified Border Element and Unified Communications Manager deployments:

- Unified Border Element: The SIP Session may be updated multiple times with corresponding media track events.
- Unified Communications Manager: If a call is on hold, the recording session is terminated. When participant resumes the call, a new recording session is created.

The following table captures this response for various representative deployment scenarios:

Deployment	Unified Border Element Forking	Unified Communications Manager Forking	Unified Border Element Forking with Unified Communications Manager Phone Endpoints
Call initiator	Participant A	Participant A	Participant A
Call receiver	Participant B	Participant B	Participant B
Hold/resume placed by	• Hold or transfer: participant A • Resume or conference: participant A	• Regardless of which participant places the hold, the recording session is terminated. • Regardless of which participant resumes the call, a new recording session is created.	• Hold (Music on Hold--M0H) or transfer: participant A • Resume or conference: participant A

Deployment	Unified Border Element Forking	Unified Communications Manager Forking	Unified Border Element Forking with Unified Communications Manager Phone Endpoints
Number of tag events received by client	Two tag events: **1** One event (Hold or Transfer): Type: TAG_EVENT, Action:ADDED, Tag name: TrackInactive, Track #: Participant B **2** One event (resume or conference): Type: TAG_EVENT, Action: ADDED, Tag name: TrackActive, Track #: Participant B	No tag events.	Six tag events: three events for hold or transfer and three events for (resume or conference): **1** Type: TAG_EVENT, Action: ADDED, Tag name: TrackInactive, Track #: Participant A **2** Type: TAG_EVENT, Action: ADDED, Tag name: TrackInactive, Track #: Participant B **3** Type: TAG_EVENT, Action: ADDED, Tag name: TrackActive, Track #: Participant A **4** Type: TAG_EVENT, Action: ADDED, Tag name: TrackInactive, Track #: Participant A **5** Type: TAG_EVENT, Action: ADDED, Tag name: TrackInactive, Track #: Participant B **6** Type: TAG_EVENT, Action: ADDED, Tag name: TrackActive, Track #: Participant B

Clients cannot delete these SYSTEM_DEFINED (TrackActive and TrackInactive) tag names at any time. The track's initial state is assumed to be active (TrackActive) by default. When the media state changes (Hold-resume or transfer-conference), the active state (TrackActive) changes to inactive (TrackInactive).

Hold-Resume and Pause-Resume

MediaSense uses "resume" to return to a call that was placed in the hold state or in the paused state:

- Resume from hold state:
    - You can do this from a device.
    - Results in the addition of a TrackActive system-defined tag event at the TRACK level.

- Resume from paused state:
    - You can do this through the APIs.
    - Results in the addition of a Resumed system-defined tag event at the SESSION level.

Correlating Recordings

Use MediaSense to correlate recordings for gateway forked media. You can correlate recordings using one of the following options:

- **Real Time correlation** provides clients with the ability to issue MediaSense API requests while the recording is active (for example, the ability to pause and resume recordings, and the ability to tag a recording with an AgentID, or with an agent-specified time-specific annotation).
- **Historical correlation** gives clients the ability to locate recordings that are selected from an external database (and vice versa) to locate call information in an external database using information from the Cisco MediaSense metadata.

To achieve this correlation, use the Call Correlation ID (CCID) and callControllerType parameters.

Use the getSessionsByCCID API to search and retrieve recorded or live sessions based on the CCID.

Correlating MediaSense Metadata with Unified Communications Manager CDR

The Cisco Unified Communications Manager Call Detail Records Administration Guide describes how to configure call detail records (CDRs) and call management records (CMRs) and provides examples of these records.

In MediaSense, when the callcontrollertype is Cisco-Unified Communications Manager , the meta data for each call only provides the xRefci (reference call ID) and the device ref of the forking device and the far-end device (can be a conference bridge or any other phone). When the callcontrollertype is Cisco-Unified Communications Manager -Gateway, metadata for each call includes a CCID (Call Correlation Identifier) field, which matches a field in the Unified Communications Manager Call Detail Recording Record.

For more information (such as the original calling number, called number, and type of call), see the Call Detail Records section in the Unified Communications Manager Call Detail Records Administration Guide available at http://www.cisco.com/en/US/products/sw/voicesw/ps556/prod_maintenance_guides_list.html.

All calls are stored on the MediaSense server.

Playing Back and Downloading Recordings

MediaSense recordings can be streamed using Real-Time Streaming Protocol (RTSP) and downloaded as an .mp4 or .wav files, or downloaded in the Raw Format using HTTP 1.1 chunked transfer coding. You can play back MediaSense recordings using any player which supports these capabilities (for example, VideoLAN Client [VLC]).

If you listen in to a forked media recording in RTSP format using some players (for example VLC), you can listen to only one track at a time (not both at the same time). If you prefer to listen to both audio channels and view the video at the same time, export any MediaSense recording to mp4 or wav format using either the mp4Url or wavUrl link. Download that file using standard HTTP access methods and then listen to both audio channels and view the video at the same time. Converting to mp4 or wav also makes the file portable and allows you to copy it to a location of your choice.

Client applications can also download the recording in its RAW Audio format by using the downloadUrl parameter returned by any of the Session Query APIs, on page 63 APIs. Each API will have a downloadUrl only for audio tracks. You cannot download MediaSense video tracks in the RAW format. The URLs (download, wav, and mp4) are conditionally present in the session query response only if the sessionState is CLOSED_NORMAL or in the sessionEvent only if the eventAction is ENDED. For other sessions in other states, (ACTIVE, DELETED, or CLOSED_ERROR), downloadUrl, wavUrl, and mp4Url are not available.

The downloadUrl provides a way to download the file using HTTP 1.1 chunked transfer coding (RFC 2616, Section 3.6.1). As such, the body contains a series of chunks that the recipient must reconstruct into the original media stream. Each chunk begins with a line containing the chunk length expressed in hexadecimal, which is followed by that exact number of bytes of binary data. The first line of the file contains the length field and a <;MEDIA-TYPE=> chunk-extension tag which indicates the type of media codec for the subsequent data. The values for the codec can be one of the following:

- "**G711-Mulaw**"
- "**G711-Alaw**"
- "**G722**"
- "**G729**"

Normally the chunks of data are directly concatenated with each other to reconstruct the media stream; however, the length line may contain an optional <;SILENCE=n> chunk-extension tag, which indicates that n milliseconds of silence should be inserted before the chunk in question. The final chunk is denoted with a length of zero bytes. All lines end with both \r and \n characters.

MediaSense also includes a "START-TIME" tag in the raw download body content immediately following the "MEDIA- TYPE" tag. START-TIME indicates when the first packet of media for each track was received by the recorder (in milliseconds since Jan 1 1970 GMT). This tag may be used by a client application to sequence and align downloaded raw media tracks so that, for example, participants do not 'talk over' each other and instead alternate with each other.

All of the URLs (including rtsp) are treated by clients as opaque strings. Client code should not depend on its format or structure in any way other than to assume that it contains a fully-formed HTTP URL, because Cisco reserves the right to alter it in the future. However, clients may add URL parameters as necessary. Specifically, adding the parameter **timeout=n** indicates that Cisco MediaSense should try for at least n seconds to write to the socket before aborting the download (the default is 5 seconds). This protects the system in case of slow networks or clients. The client can determine whether the download ran to completion by confirming that the last line received ends with "0\r\n".

Following is an example of a G711-Mulaw media stream, which is divided into three chunks, for a total of 26796 bytes of data, with two interspersed silence segments totaling 240ms.

```
------------
1234;MEDIA-TYPE='G711-mulaw';START-TIME=2069539211\r\n
0x1234 bytes of binary data\r\n
2222;SILENCE='40'\r\n
0x2222 bytes of data\r\n
3456;SILENCE='200'\r\n
0x3456 bytes of binary data\r\n
7788\r\n
0x7788 bytes of binary data\r\n
0\r\n
```

The rtspUrl parameter allows streaming while the downloadUrl, wavUrl, and mp4url parameters provide downloading capability. Consequently, the rtspURL is provided for both active and closed sessions, while the other URLs are only available for closed sessions.

RTSP and HTTP Request Authentication and Redirection

MediaSense uses basic authentication to validate the username and password of API clients that make Real-Time Streaming Protocol (RTSP) requests and media-related HTTP requests. The OPTIONS command is an exception to this pattern because it does not require authentication; it may, however, be redirected.

After authentication, MediaSense redirects RTSP requests and media-related HTTP requests to another URL. Do not parse or cache the redirected URL because it is opaque and is subject to change.

Following are examples of RTSP authentication and redirection of the DESCRIBE command:

- Before authentication
- With authentication
- After redirection

DESCRIBE (with Authentication)

```
REQUEST
DESCRIBE rtsp://10.194.118.94/archive/e4137a336b0bf1 RTSP/1.0
CSeq: 7
Authorization: Basix YXBpdXNIcjpaXNjbw==
User-Agent: LibVLC/1.1.9 {LIVE555; Streaming Media v2011.03.14}
Accept: application/sdp

RESPONSE
RTSP/1.0 302 Moved Temporarily
Server: Cisco MediaSense Media Server
CSeq: 7
Location: rtsp://10.194.118.94:9554/archive/e4137a336b0bf1?token=abc123
```

DESCRIBE (Before Authentication)

```
REQUEST
DESCRIBE rtsp://10.194.118.94/archive/e4137a336b0bf1 RTSP/1.0
CSeq: 6
User-Agent: LibVLC/1.1.9 {LIVE555; Streaming Media v2011.03.14}
Accept:  application/sdp

RESPONSE
RTSP/1.0 401 Unauthorized
WWW-Authenticate: Basic realm="Secured Area"
CSeq: 6
Server: Cisco MediaSense Media Server
```

DESCRIBE (After Redirection)

```
REQUEST
DESCRIBE rtsp://10.194.118.94:9554/archive/e4137a336b0bf1?token=abc123 RTSP/1.0
```

```
CSeq: 8
User-Agent: LibVLC/1.1.9 (LIVE555; Streaming Media v2011.03.14)
Accept: application/sdp

RESPONSE
RTSP/1.0 200 ok
CSeq: 8
Content-Type: application/sdp
Content-Length=512

V=0
O=15237780159166911470 15237780159166912070 IN IP4 10.194.118.94
a=Cisco Live Media Streaming Session
```

Distributing HTTP Download Requests

Some clients use the HTTP Download facility to create copies of all recordings, using MediaSense more as a temporary location for these files than as a long-term archive. Using the download facility is a perfectly valid use case. However, you may be tempted to batch up these download requests and issue them once a day (or on some other periodic basis).

Distribute these requests evenly over time to make better use of your resources. For example, use the session ENDED event to trigger a download as soon as the call recording terminates.

Media Forking

All MediaSense-supported Cisco IP phones have a built-in bridge (BIB), which allow incoming and outgoing media streams to be forked. MediaSense makes use of this capability to record inbound and outbound forked media. For more details about media forking, see the Unified Communications Manager documentation.

Unified Border Element does not have a concept of BIB as the call forking is performed within the Unified Border Element application—not from a phone.

Cisco Unified Communications Manager Network-based Recording

With Unified Communications Manager network-based recording (NBR), you can use a gateway to record calls. NBR allows the Unified Communications Manager to route recording calls, regardless of device, location, or geography.

With NBR, call recording media can be sourced from either the IP phone or from a gateway that is connected to the Unified Communications Manager over a SIP trunk. Unified Communications Manager dynamically selects the right media source based on the call flow and call participants.

NBR offers an automatic fallback to Built-in-Bridge (BiB) when the Integrated Services Routers (ISR) are unavailable as no separate recording configuration is required. This is useful in cases where customers want to include agent-agent consult calls in the recording policies as Unified Border Element cannot record consult calls, so BiB needs to be enabled separately.

For more information on Unified Communications Manager NBR, refer to *Features and Services Guide for Cisco Unified Communications Manager* at http://www.cisco.com/c/en/us/support/unified-communications/unified-communications-manager-callmanager/products-maintenance-guides-list.html

Note MediaSense supports TDM gateway recording.

Blog Recording

MediaSense enables you to create blog recordings (audio and video) using supported Cisco IP phones. After you record blog recordings, third-party applications can publish them.

A blog recording is initiated in one of the following ways:

- By a user who dials into a MediaSense server.
- By the MediaSense server calling a user's phone in response to an API request.

Cluster Deployment

MediaSense servers are deployed in a cluster. A cluster can contain one to five servers. Depending on your deployment, each cluster can provide basic media recording and database storage, and handle scalable recording capacity.

- One-server clusters are deployed by small businesses. In this case, the server manages the media recordings, the database storage, and the configuration information.
- Two-server clusters are deployed by businesses concerned about data replication and redundancy. In this case, both servers manage the media recordings, the database storage, and the configuration information. This deployment provides High Availability and ensures that all database data is stored redundantly.
- Three- to five-server clusters provide greater recording capacity and handle a heavier load. The expansion servers manage the additional media recordings.

Differences Between Unified Communications Manager and Unified Border Element Scenarios

Unified Communications Manager is used to set up the recording profile or call control service connection (SIP trunk) with MediaSense. Similarly, with Unified Border Element, the dial peers and media class settings determine communication with MediaSense. Almost everything not related to call signaling is the same between Unified Communications Manager scenarios using MediaSense and Unified Border Element scenarios using MediaSense. Regardless of MediaSense being deployed with Unified Communications Manager or Unified Border Element, events, response codes, and parameter definitions are the same for both scenarios.

One of the API (application programming interface) parameters used by MediaSense to distinguish between call provider types is the callControllerType parameter. While this parameter is present in both deployments, the value of the parameter differs for Unified Border Element and Unified Communications Manager.

The following table highlights the major API-related differences between both scenarios.

MediaSense Feature	With Unified Communications Manager (Applies to both Built-in-Bridge Recording and Network-based Recording)	With Unified Border Element Dial Peer Forking
Assignment of the trackNumber parameter	SessionEvents and sessions (in the API responses) have the following track assignment: • Track 0 is associated with one participant (forking phone). • Track 1 is associated with one participant (non-forking phone) or multiple participants (if non-forking phone transfer call).	The trackNumber assignment is arbitrary and is based on whether the call is incoming or outgoing from the dial peer on which media forking is enabled.
Call correlation (xRefCi parameter)	The xRefCi parameter is generated by Unified Communications Manager.	The xRefCi parameter is generated by MediaSense. In this case, you will not see the xRefCi details in any Unified Communications Manager record. The MediaSense solution allows you to correlate recordings for gateway forked media. To achieve this correlation, use the Call Correlation ID (CCID) and callControllerType parameters. For more information, see the Correlating Recordings section. The CCID parameter corresponds to the Cisco GUID generated by Unified Border Element and can be used for solution-wide end-to-end call trace.
Identifying tracks for calling versus called party. For more information, see the FAQs for Cisco MediaSense website (How do you determine which track has the calling and which has the called party?).	The numerically smaller xRefCi parameter almost always refers to the calling party's track.	Track 0 contains the media stream corresponding to the dial peer in which the media recording profile is configured.

MediaSense Feature	With Unified Communications Manager (Applies to both Built-in-Bridge Recording and Network-based Recording)	With Unified Border Element Dial Peer Forking
Call placed on hold	• When a call is place on hold, the logical recording session is terminated. Sessions are related—they share the same pair of xRefCi parameters. • The session is terminated for the near end.	• The SIP Session may be updated multiple times with corresponding media track events. Only one session despite any hold/resume sequences. • Adds the SYSTEM_DEFINED TrackInactive tag when you place a call on hold.
Call resumed	• When a participant resumes the call, a new recording session is created. Sessions are related—they share the same pair of xRefCi parameters. • A new session is started for the near end.	• The SIP Session may be updated multiple times with corresponding media track events. Only one session despite any hold/resume sequences. • Adds the SYSTEM_DEFINED TrackActive tag when you resume the call.

Note For more information about other deployment and administration-related differences, see the User Guide for Cisco MediaSense (available at http://www.cisco.com/en/US/products/ps11389/products_user_guide_list.html).

CHAPTER 2

Working with Cisco MediaSense APIs

MediaSense API Conventions

MediaSense APIs use the camelCase convention. The API URIs are case sensitive. Use the exact URI as identified for each API. The parameters are not case sensitive.

Using HTTPS

MediaSense APIs only support HTTPS. Unsecured HTTP is unaccepted for most MediaSense APIs. The only exception is getAPIVersion that can accept either HTTP or HTTPS.

MediaSense uses HTTP version 1.1, and attempts to keep client connections in place from one request to the next. To ensure better performance, especially for event delivery, be sure to design third-party applications to do the same.

Using POST or GET Requests

MediaSense APIs use the following conventions for POST and GET requests:

HTTP Request Type	Description	Action	Example
POST	Any request that performs an action on the server and changes the state of the MediaSense server (start or stop requests). Or any query or read request in which the parameter structure is too complex to specify with query parameters. For example, the getSessions API uses POST even though it is a *safe method* (does not change the state of the server).	Parameters are passed in the POST data block. The data block uses JSON syntax.	`{ "requestParameters":{ "param1":"value1", "param2":"value2" } }`
GET	Any request that only queries or reads information (intended only for information retrieval and does not change the state of the server, safe methods) from the MediaSense system.	Parameters are passed in the URL as query parameters. When specifying query parameters, append a question mark "?" to the first parameter, and an ampersand "&" to each of the subsequent parameters.	`http://cisco.com/abc /xyz?param1=value1 &param2=value2`

API Version

MediaSense APIs follow the version conventions:

- The MediaSense Release 11.0(1) API version is 1.7.
- The API version number starts at 1.0 and is not tied to the product version.
- The API version number is x.y and not x.y.z (for example, 1.0).

Use the getAPIVersion API to retrieve the current version of the APIs running on the system. This API returns a version number for the APIs running in any MediaSense deployment. You do not need to sign in to use this API.

JSON Format in Responses

MediaSense supports JSON. API responses and events are in the JSON format. Regardless of the format used in your third-party applications that use MediaSense APIs, follow these conventions:

- Specify content type as either application or json in the HTTP header of each API request.
- If the header is missing, the data is returned in the default JSON format.

Note The JSON format is a key-value pair. The order of the key-value pairs may not be the same in each response.

Cisco usually adds new properties to the JSON response schema from one API version to the next, but does not modify either the meaning or the relative position in the object hierarchy of any existing property. In order to ensure your client code functions properly with newer versions of MediaSense, you must ignore any JSON properties you do not recognize. This applies to both API responses and events.

Key Elements for MediaSense APIs

This section identifies the key elements and pattern within a MediaSense API.

Sample URI for a MediaSense API.

https://<host>:<port>/ora/<sampleService>/<sample>/<sampleMethod>

The breakdown of this sample URI is explained in the following table:

Element	Description
https://	The secure protocol used by MediaSense APIs. • Server-server communications must always use HTTPS. • Client-server communications must use HTTPS for authentication. • HTTPS is required for almost all MediaSense APIs. The one exception is getAPIVersion API, which can take either HTTP or HTTPS.
<host>:<port>	Replace <host> with the hostname and <port> with the port number. These two elements are part of the context root for each API.
ora	The product name specification for MediaSense APIs. This element remains constant throughout all MediaSense APIs.
sampleService or sample	The name of the service being addressed by a particular MediaSense API. Some of the available services are User Authentication, Event Subscription, Session Query, Session Management, and so forth. In this element, **sampleService** is the interface and **sample** is its implementation.
sampleMethod	The API name. Each API name begins with a verb to indicate the action being performed.

The following URI is an example of the MediaSense subscribeRecording API.

https://<host>:<port>/ora/eventService/event/subscribeToEvents

API Response Schema

All MediaSense API responses have these key elements.

JSON schema:

```
{
"responseCode": <replace with your integer> //Numeric code for the response
"responseMessage": <replace with your string> //A textual description of the result
"responseBody": {<JSON object>} //The response itself (optional)
}
```

Note The responseMessage parameter is subject to change and is not intended to be used for programmatic comparison purposes. Instead, use the responseCode parameter for this purpose.

Asynchronous Event Schema

All MediaSense API responses have these key elements.

JSON example:

```
{
    "eventType": <string value>, //type of the event, such as SESSION_EVENT
    "eventAction": <string value>, //possible actions for a given event such as
SESSION_STARTED
    "forwardedEvent": <true or false>, //indicates that the event is not locally generated
    "eventBody": <JSON object> //the event itself
}
```

Note See Shared Parameters, on page 112 for details about each of these elements.

API Response Codes

2xxx Success Response Codes

The action was successfully received, understood, and accepted.

Response Code	Description
2000	Success: Your request was successfully completed.
2001	Success: No results found for this client request.
2002	Success: The session recording was successfully paused but the database could not be updated.
2003	Success: The session recording was successfully resumed but the database could not be updated.
2004	Success: The session recording was successfully deleted but the database could not be updated.
2005	Success: The subscription with the given subscriptionUri already exists.
2006	Success: The current subscription has been updated and has been unsubscribed from the subscriptionFilters requested. The subscription is still active.

3xxx Redirection Response Code

The client must take additional action to complete the request.

Response Code	Description
3001	Failure : Unable to process the request. Provide redirection and try again.

4xxx Client Error Response Codes

The request contains bad syntax or cannot be fulfilled.

Response Code	Description
4000	Failure: Your request is invalid.
4001	Failure: Unsecured HTTP is not allowed. Use HTTPS for your request.
4005	Failure: Cannot find value in database. Detail: <parameter name>: <parameter value>
4019	Failure: Not an authorized MediaSense user.
4020	Failure: Your login credentials (username or password) are invalid.
4021	Failure: Invalid session. The session may have expired. Sign in again or enter a valid JSESSIONID.
4022	Failure: The specified user is not a Finesse supervisor.
4023	Failure: Connection to the Cisco Finesse server failed. Verify the Cisco Finesse configuration and try again.
4030	Failure: Unable to establish a call using this deviceRef. Verify the deviceRef and try again.
4031	Failure: The deviceRef is unavailable or busy. Verify the device and try again later.
4032	Failure: Unable to disconnect the call using this deviceRef. Verify if the call exists and try again.
4033	Failure: Unsupported combination of mediaStreams. Verify the allowed streams for this parameter and try again.
4041	Failure: The fieldName parameter is invalid.
4042	Failure: The fieldOperator parameter is invalid.
4043	Failure: The number of fieldValue parameters is invalid.
4045	Failure: The connector parameter is invalid.
4046	Failure: The order parameter is invalid.
4047	Failure: The fieldName parameter is missing.
4048	Failure: The fieldCondition parameters are missing.
4049	Failure: The fieldOperator parameter is missing.
4050	Failure: The fieldValue parameters are missing.

Response Code	Description
4051	Failure: The fieldConnector parameter is missing.
4052	Failure: The paramConnector parameter is missing.
4053	Failure: The fieldName parameter is missing. Unable to sort by fieldName.
4054	Failure: The order parameter is missing.
4055	Failure: The requestParameter is missing.
4056	Failure: Invalid JSON syntax. Verify the syntax and try again.
4057	Failure: One or more of the fieldValue parameters are of the incorrect type.
4058	Failure: Missing tagName parameter.
4059	Failure: The authentication provider is invalid. Valid authentication providers are: AXL, FINESSE
4060	Failure: The value is missing in the client request.
4061	Failure: Missing parameter in message. Detail: <parameter name>
4062	Failure: Invalid syntax in parameter. Detail: <parameter name>
4063	Failure: The offset parameter is invalid.
4064	Failure: The limit parameter is invalid.
4066	Failure: The date range is invalid. Date range of up to one year is allowed. Resubmit your request.
4070	Failure: The sessionId is invalid or no active session is found.
4071	Failure: The sessionId is invalid or no inactive session is found.
4072	Failure: The sessionId is invalid or no session is found.
4073	Failure: The conversionFormat is invalid. Please verify the supported formats and try again.
4080	Failure: The subscription with the given subscriptionUri already exists.
4081	Failure: The subscription with the given subscriptionId does not exist.
4082	Failure: Unable to convert a session that is in the ACTIVE or ERROR state.
4090	Failure: The jobId is invalid or particular job is not running at this moment.

5xxx Server Error Response Codes

The server failed to fulfill an apparently valid request.

Response Code	Description
5000	Failure: An unknown server error occurred. Try again later.
5001	Failure: A database error occurred. Try again later.

Response Code	Description
5002	Failure: Unable to pause the session. Try again later.
5003	Failure: Unable to resume the session. Try again later.
5004	Failure: Unable to delete the session. Try again later.
5006	Failure: Unable to convert the session. Try again later. Detail: <Reason For Failure>
5020	Failure: AXL service on the Unified Communications Manager timed out. Try again later.
5021	Failure: AXL service on the Unified Communications Manager is not activated. Activate the service and try again.
5022	Failure: AXL credentials are invalid. Verify the Unified Communications Manager configuration and try again.
5023	Failure: Unknown AXL Host. Verify the Unified Communications Manager configuration and try again.
5024	Failure: Unknown AXL service error. Try again later.
5025	Failure: Connection to the AXL service failed. Try again later.
5030	Failure: Unable to establish a call using this DeviceRef in association with the system. Verify the Unified Communications Manager or Unified Border Element system and try again.
5031	Failure: Unable to record the call. Verify that the MediaSense Media Service is functioning and try again.
5032	Failure: Unable to record the call. Low disk space on MediaSense Media Service.
5040	Failure: Unable to process the request. Invalid state of the job for the requested operation. Verify the job state and try again. Detail: <Job State>
5041	Failure: Request timed out. Resubmit your request.
5042	Failure: The system has exceeded the maximum number of concurrent requests. Try again later.
5053	Failure: Unable to process the request. The maximum limit to search the archived sessions has exceeded. Narrow down your search.

Response Message

The response message can have the following types of values:

- Positive response: "Success" or "Partial success".
- Negative response: "Failure: <text indicating reason for failure>".

Response Body

The API response can also contain an optional body based on requirements (responseBody). Any information other than the two preceding keys is placed within the responseBody.

JSON schema:

```
"responseBody": {//one or more optional elements within the
 responseBody depending on the request specifications.
          ..
                        ..
        }
```

Note All parameters may not be present in the response body of an API as these vary on a case-to-case basis. In these cases, only the applicable parameters appear in the response body of the related APIs. For example, in the response body of the getSessions API, an active session will not contain the duration parameter details.

Encoding

All URLs must be URL encoded (Percent-encoding).

Special Characters in Text Strings

For all text strings (including passwords), users must abide by the http://www.json.org/ specification and escape any special characters by preceding them with a backslash.

Job States

Job States	Description
RUNNING	The job started execution successfully and continues to execute.
COMPLETED	The job completed successfully. **Note** Completed successfully does not mean all operations are successful, it just indicates that all operations in this job were attempted successfully.
CANCELED	The job was canceled using the cancelJob API and is no longer running.
ERROR	The job stopped executing because of a system error.

Precedence Rules for paramConnector and fieldConnector

The query syntax for getSessions allows flexibility (though there are limitations; not every conceivable query can be expressed). A query is formed by linking terms together by AND and OR operator; each term is made up of a field name, a relational operator, and a field value (or two field values in case of BETWEEN). To introduce parentheses follow the below mentioned precedence rules that direct the order in which the terms are evaluated.

- Multiple conditions on the same field get the highest logical precedence.
- AND takes precedence over OR.

For example:

For query `deviceRef=1000 or deviceRef=2000 or tagName=foo and state=active`, the highest precedence is given to the first two terms even though they are linked by OR because they refer to the same field name.

Next, tagName=foo and state=active are evaluated because AND has higher priority than OR.

Finally the results of the two previous evaluations are calculated as OR with unlike field names has the lowest priority.

If the above query is represented with parentheses, then it looks like:

```
((deviceRef=1000 or deviceRef=2000) or (tagName=foo and state=active))
```

The above example is written in a quasi-English form in order to be readable for the purpose of explanation. But in the actual API request, it would be represented in the JSON form. For a detailed example, see the getSessions, on page 76.

Encoding

All URLs must be URL encoded (Percent-encoding).

Special Characters in Text Strings

For all text strings (including passwords), users must abide by the http://www.json.org/ specification and escape any special characters by preceding them with a backslash.

Request and Response Parameter Definitions

For a list of request and response parameters used by the MediaSense API, see Shared Parameters, on page 112.

Failover Between Two MediaSense Servers

If your MediaSense cluster contains multiple servers, only two of these servers provide the API service that handle the APIs. MediaSense does not support internal redirection of client requests between the nodes and hence does not support failover. If the MediaSense API service that handles these REST requests is out-of-service for any reason, it sends a response code that signals the third-party client to redirect the request to the other MediaSense server providing this service. The client interprets this response code and redirects the request accordingly.

If the MediaSense API service is down or the server itself is down, then the client must handle the HTTP timeout and redirect the request to the other MediaSense server accordingly.

Security Considerations

All MediaSense APIs are based on HTTPS and use self-signed certificates. You may see a security exception each time you use an API.

A secure server uses a certificate to identify itself to web browsers. You can generate your own certificate (called a self-signed certificate) or you can obtain a certificate from a certificate authority (CA). For more information, see http://en.wikipedia.org/wiki/Certificate_authority or your web browser documentation.

Note If you use Poster or other similar applications to issue the API requests, you must first obtain a certificate by opening and accepting the following URI in your web browser:

https://<Cisco MediaSense IP address>>:8440

API Inter-Dependencies for Authentication

MediaSense APIs use the JSESSIONID to maintain all sessions, therefore a JSESSIONID is required to authenticate the user before any other API (that requires authentication) can be used.

Use the MediaSense SignIn API to obtain a JSESSIONID. The response from the SignIn API looks like this:

```
Header Name: Set-Cookie
Header Value: JSESSIONID=SomeRandomAlphaNumericString
```

Each time you use a MediaSense API that requires authentication, you must set the header to the value of the JSESSIONID returned from the SignIn API.

For more information about setting cookies, see http://en.wikipedia.org/wiki/HTTP_cookie.

The JSESSIONID expires when the user logs out or after 30 minutes of inactivity, whichever comes first.

Poster

All MediaSense APIs are accessible by a URI and follow a request-response paradigm. Poster is a Firefox add-on that works with web services and web resources so you can interact with web services and inspect the results.

To add Poster to Firefox:

- Configure a MediaSense API user (using a valid username and password).
- Download the Poster add-on from Firefox. You can obtain a free download from https://addons.mozilla.org/en-US/firefox/addon/2691/.

- After you add Poster to Firefox, type **Ctrl** + **Alt** + **P** to launch it.

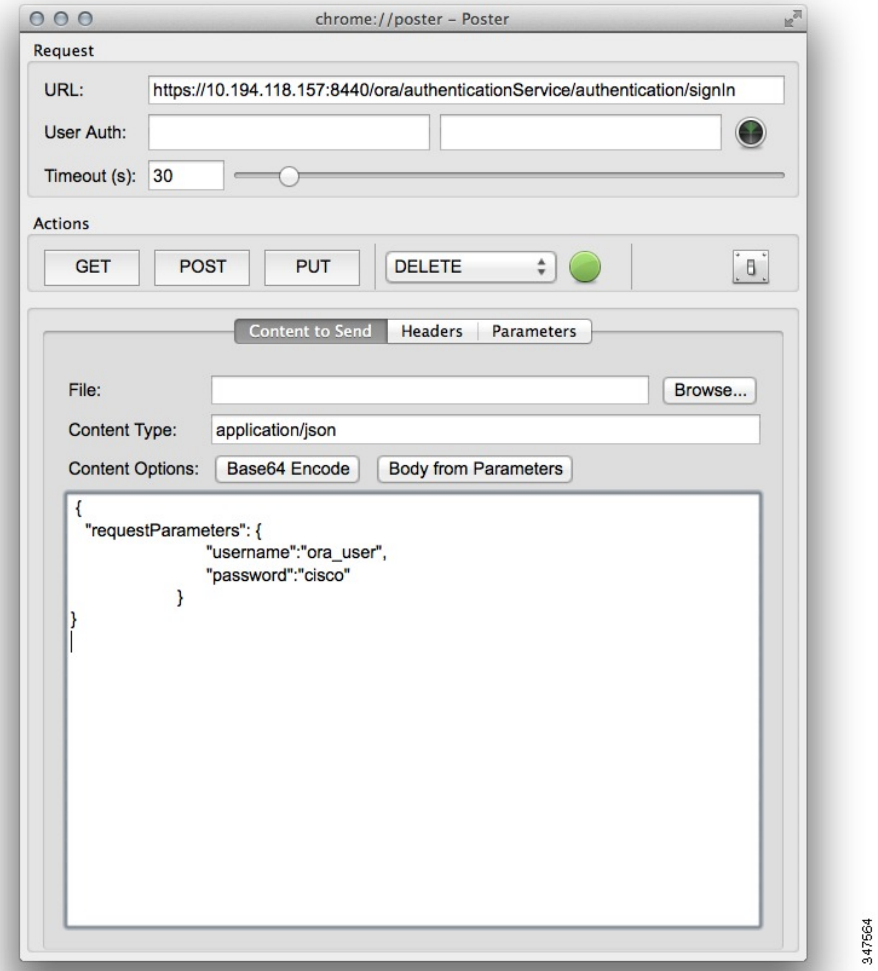
To test an API in Poster:

- Accept the MediaSense security certificate into the web browser by opening and accepting the following URI in the web browser.

  https://<Cisco MediaSense IP address>>:8440. You have to do this one time.
- Copy and paste the URI for the API request from this Developer Guide into a text editor. For example, to enter the URI for signing in, copy the URI from the signIn API.
- Examine the pasted code for case sensitivity and format and to remove any carriage returns.
- Update the URI with the IP Address and port of your MediaSense server.
- Add any mandatory parameters for the request.
- Set the content-type header with the appropriate value.

- Click the appropriate action (GET or POST).

Figure 2: A Cisco MediaSense API Request Using Poster



URL: https://10.194.118.157:8440/ora/authenticationService/authentication/signIn

User Auth:

Timeout (s): 30

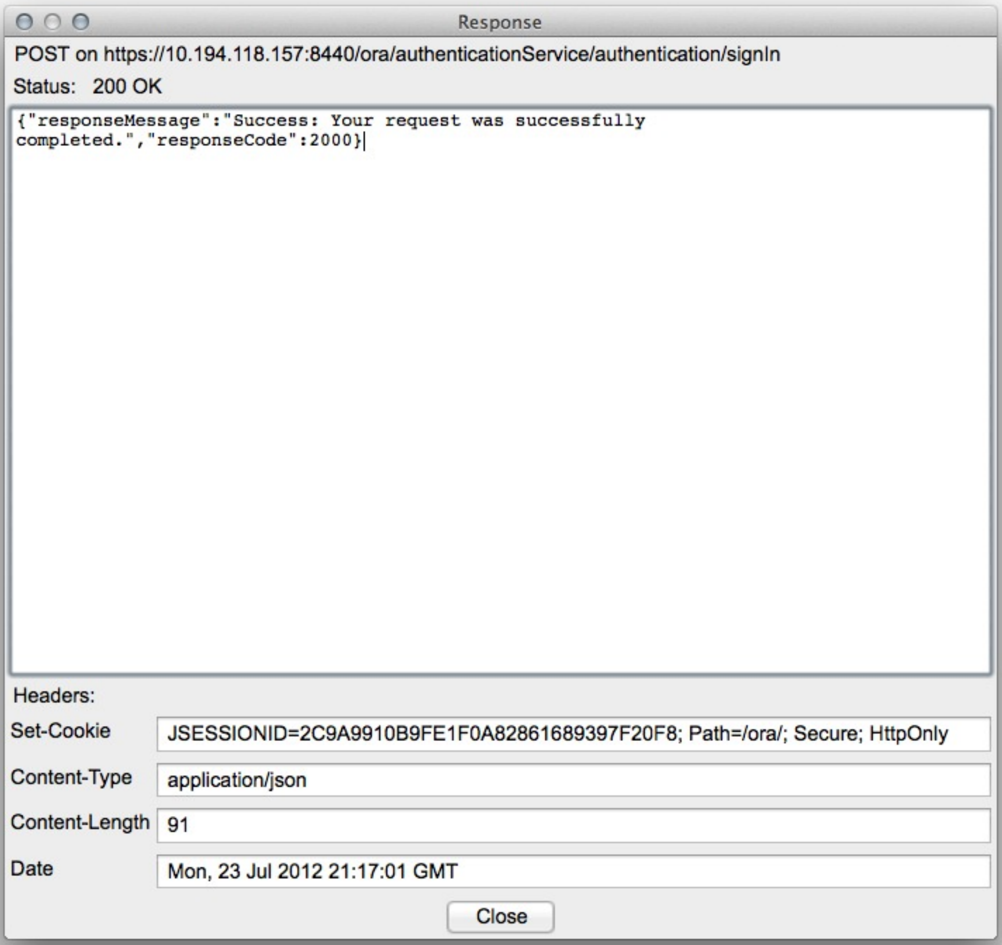
File:

Content Type: application/json

Content Options:

```
{
  "requestParameters": {
                       "username":"ora_user",
                       "password":"cisco"
                  }
}
```

Figure 3: A Cisco MediaSense API Response Status Using Poster



POST on https://10.194.118.157:8440/ora/authenticationService/authentication/signIn

Status: 200 OK

```
{"responseMessage":"Success: Your request was successfully
completed.","responseCode":2000}
```

Headers:

Set-Cookie	JSESSIONID=2C9A9910B9FE1F0A82861689397F20F8; Path=/ora/; Secure; HttpOnly
Content-Type	application/json
Content-Length	91
Date	Mon, 23 Jul 2012 21:17:01 GMT

347563

CHAPTER 3

Event Subscription APIs

Introduction

The eventSubscription APIs allow you to subscribe, verify subscription, and unsubscribe for various event notifications.

subscribeToEvents

Use this API to receive event notifications, such as when recording starts, recording ends, recording data gets updated, a tag is added to the recording, or when a tag is deleted from the recording. You may choose to subscribe to all events, categories of events or specific types of events.

URI

```
https://<host>:<port>/ora/eventService/event/subscribeToEvents
```

HTTP Method

POST

Parameters

- subscriptionFilters— It is an output JSON array and an optional input JSON array. It specifies a list of events or a list of event categories. See Shared Parameters, on page 112.

- subscriptionId— It is an output string. See Shared Parameters, on page 112.
- subscriptionType— It is an output string. It specifies a type of subscription. See Shared Parameters, on page 112.
- subscriptionUri— It is the required input string. It specifies the URI where the event notifications are sent to server-based clients. See Shared Parameters, on page 112.
- tagNameRegEx— It is an optional input string. It is a Java regular expression that only applies to tag events. See Shared Parameters, on page 112.

Note If the same subscriptionUri is subscribed for a second time, the subscriptionFilters in the existing subscription will be replaced with the new subscriptionFilters. If the subscriptionFilters are also the same, then no error is generated. A 2005 responseCode is generated indicating that the subscription already exists.

Examples

Example 1

To receive event notifications for all events.

HTTPS POST:

```
https://10.194.118.1:8440/ora/eventService/event/
subscribeToEvents
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   requestParameters: {
                         "subscriptionType": "http",
                         "subscriptionUri": "http://10.35.146.157:
                          8085/sessionEvent",
                         "subscriptionFilters": ["ALL_EVENTS"]
                       }
}
```

Response:

```
{
    "responseMessage": "Success: Your request was successfully
     completed.",
    "responseCode": 2000,
    "responseBody": {
        "subscriptionFilters": [
            "EXIT_EMERGENCY_STORAGE_SPACE_EVENT",
            "TAG_ADDED_EVENT",
            "TAG_DELETED_EVENT",
            "TAG_UPDATED_EVENT",
            "SESSION_STARTED_EVENT",
            "SESSION_ENDED_EVENT",
            "EXIT_LOW_STORAGE_SPACE_EVENT",
            "EXIT_CRTITICAL_STORAGE_SPACE_EVENT",
            "ENTER_CRTITICAL_STORAGE_SPACE_EVENT",
            "ENTER_LOW_STORAGE_SPACE_EVENT",
            "SESSION_UPDATED_EVENT",
            "SESSION_DELETED_EVENT",
            "ENTER_EMERGENCY_STORAGE_SPACE_EVENT",
            "SESSION_PRUNED_EVENT"
        ],
        "subscriptionId": "3oV6jCEnJUlGYZo8"
    }
}
```

Example 2

To receive event notifications for only**TAG_EVENTS**.

HTTPS POST:

```
https://10.194.118.1:8440/ora/eventService/event/
subscribeToEvents
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   requestParameters: {
                        "subscriptionType": "http",
                        "subscriptionUri": "http://10.35.146.157:
                         8085/sessionEvent",
                        "subscriptionFilters": ["TAG_EVENTS"]
                      }
}
```

Response:

```
{
    "responseMessage": "Success: Your request was successfully
     completed.",
    "responseCode": 2000,
    "responseBody": {
        "subscriptionFilters": [
            "TAG_ADDED_EVENT",
            "TAG_DELETED_EVENT",
            "TAG_UPDATED_EVENT"
        ],
        "subscriptionId": "D52C4aeXISTURzM7"
    }
}
```

unsubscribeFromEvents

Use this API to stop receiving recording events.

- A request without a "subscriptionFilters" parameter unsubscribes the client completely and terminates the subscription.
- A request with a "subscriptionFilters" parameter only removes the specified filter from the subscription, but the subscription as a whole remains active. If no filters remain subscribed, then the entire subscription is removed, exactly as if no "subscriptionFilters" parameter were specified.
- The response also includes a list of the remaining subscription filters, unless there are no event types left.

URI

```
https://<host>:<port>/ora/eventService/event/unsubscribeFromEvents
```

HTTP Method

POST

Parameters

- subscriptionId— It is a required input string. See Shared Parameters, on page 112.
- subscriptionFilters— It is an output JSON array and an optional input JSON array. It specifies a list of events or a list of event categories. See Shared Parameters, on page 112.
- tagNameRegEx— It is an optional input string. It is a regular Java expression which only applies to tag events. See Shared Parameters, on page 112.

Examples

Example 1

To stop receiving event notifications for any event (unsubscribe from all events):

HTTPS POST:

```
https://10.194.118.1:8440/ora/eventService/event/
unsubscribeFromEvents
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   requestParameters: {
                        "subscriptionId": "sN7DASoAArHmDfI1",
                        "subscriptionFilters":[ALL_EVENTS]
                      }
}
```

OR

```
{
   requestParameters: {
                        "subscriptionId": "sN7DASoAArHmDfI1"
                      }
}
```

Response:

```
{
    "responseCode": 2000,
    "responseMessage": "Successful"
}
```

Example 2

The client is currently subscribed to RECORDING_EVENTS and CLEANUP_EVENTS and now wants to unsubscribe from CLEANUP_EVENTS, but continue to maintain their subscription to RECORDING_EVENTS.

HTTPS POST:

```
https://10.194.118.1:8440/ora/eventService/event/
unsubscribeFromEvents
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   requestParameters: {
                        "subscriptionId": "sN7DASoAArHmDfI1",
                        "subscriptionFilters":[CLEANUP_EVENTS]
                      }
}
```

Response: (This will tell the client the filters they are still subscribed to.)

```
{
    "responseMessage": "Success: Your request was successfully
     completed.",
    "responseCode": 2000,
    "responseBody": {
        "subscriptionFilters": [
            "SESSION_STARTED_EVENT",
            "SESSION_UPDATED_EVENT",
            "SESSION_ENDED_EVENT"
        ]
    }
}
```

verifyEventSubscription

Use this API to verify if the subscription to receive events is active or not.

URI

```
https://<host>:<port>/ora/eventService/event/verifyEventSubscription
```

HTTP Method

POST

Parameters

- subscriptionFilters— It is an output JSON array and an optional input JSON array. It specifies a list of events or event categories. See Shared Parameters, on page 112.
- subscriptionId— It is a required input string. It is a system-generated ID. See Shared Parameters, on page 112.
- subscriptionStatus— It is an output string. It is a response received from the API. See Shared Parameters, on page 112.
- tagNameRegEx— It is an optional input string. It is a Java regular expression that only applies to tag events. See Shared Parameters, on page 112.

Examples

Example 1

Assuming that you are currently subscribed to receive all RECORDING_EVENTS, use the following API to verify that your subscription is active:

HTTPS POST:

```
https://10.194.118.1:8440/ora/eventService/event/
verifyEventSubscription
```

Headers:

```
{
   requestParameters: {
                        "subscriptionId": "sN7DASoAArHmDfI1"
                      }
}
```

Response:

```
{
    "responseMessage": "Success: Your request was successfully
     completed.",
    "responseCode": 2000,
    "responseBody": {
        "subscriptionFilters": [
            "SESSION_STARTED_EVENT",
            "SESSION_UPDATED_EVENT",
            "SESSION_ENDED_EVENT"
        ],
        "subscriptionStatus": "ACTIVE"
    }
}
```

Example 2

Verify an INACTIVE subscription.

HTTPS POST:

```
https://10.194.118.1:8440/ora/eventService/event/
verifyEventSubscription
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   requestParameters: {
                        "subscriptionId": "sN7DASoAArHmDfI2"
                      }
}
```

Response:

```
{
    "responseMessage": "Success: Your request was successfully
     completed.",
    "responseCode": 2000,
    "responseBody": {
        "subscriptionStatus": "INACTIVE"
    }
}
```

subscribeRecordingEvent(deprecated)

Use this API to receive event notifications, such as when recording starts, recording ends, recording data gets updated, a tag is added to the recording, or when a tag is deleted from the recording.

URI

```
https://<host>:<port>/ora/eventService/event/subscribeRecordingEvent
```

HTTP Post

POST

Parameters

- subscriptionId— It is a required input string. It is the system-generated ID. See Shared Parameters, on page 112.
- subscriptionType— It is an output string. It is the type of subscription. See Shared Parameters, on page 112.
- subscriptionUri— It is a required input string. It is the URI where event notifications are sent to server-based clients. See Shared Parameters, on page 112.

Note If the same subscriptionUri is subscribed for a second time, an error will be returned and it will contain the original subscriptionId.

Related Event

storageThresholdEvent— The event is sent each time the storage disk space reaches various thresholds. For more information, see storageThresholdEvent, on page 134.

unsubscribeRecordingEvent(deprecated)

Use this API to stop receiving recording events.

URI

```
https://<host>:<port>/ora/eventService/event/unsubscribeRecordingEvent
```

HTTP Method

POST

Parameter

subscriptionId— It is a required input string. It is the system-generated ID. See Shared Parameters, on page 112

verifyRecordingSubscription(deprecated)

Use this API to verify if the recording subscription is active.

URI

```
https://<host>:<port>/ora/eventService/event/verifyRecordingSubscription
```

HTTP Method

POST

Parameter

- subscriptionId— It is a required input string. It is the system-generated ID. See Shared Parameters, on page 112.
- subscriptionStatus— It is an output string. It is the response that is received from the API. See Shared Parameters, on page 112.

Note The subscriptionStatus is active or inactive depending upon whether the subscription exists or not.

CHAPTER 4

Job Management APIs

Introduction

A job can contain one or more operations and takes a significant amount of time to complete. Each result is provided on a per-operation basis. The purpose of a job is to allow the system to accept and act on the job request without requiring a persistent client connection to the server until the job is completed.

You can create a MediaSense job by using the deleteSessions API, which creates a job in response to a bulk delete operation, where you may delete multiple sessions at the same time.

You can only delete a job that has already been completed, canceled, or in an error state. If the job is in the RUNNING state, use the cancelJob API to first stop the job and then use the deleteJob API to delete that job.

A job may not eventually be acted upon, as it might be disallowed during the processing phase. Use the Job Query APIs to retrieve the job status and operation results.

createJob

Use this API to create a batch process for jobs which take a significant amount of time to complete. The results are provided for each operation.

URI

```
https://<host>:<port>/ora/managementService/manage/createJob
```

HTTP Method

POST

Parameters

- jobType— It is an input string. The type of the job. See Shared Parameters, on page 112.
- jobParameters— It is a system-generated identifier for a session. See Shared Parameters, on page 112.

Note See Job States, on page 20.

cancelJob

Use this API to cancel a job that is already in progress. This API only stops the execution of a job. To delete the job from the database, use the deleteJob API.

URI

```
https://<host>:<port>/ora/managementService/manage/cancelJob
```

Parameter

jobId— It is an output string. This is a system-generated job ID. See Shared Parameters, on page 112.

HTTP Method

POST

Example

HTTPS POST:

```
https://10.194.118.1:8440/ora/managementService/manage/cancelJob
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
 "requestParameters": {
 "jobId": "123456789"
 }
}
```

Response:

```
{
 "responseMessage": "Success: Your request was successfully completed.",
 "responseCode": 2000,
 "jobId": "123456789"
}
```

deleteJob

Use this API to delete a job from the database. This API is only applicable to a job that is already completed, canceled, or in an error state. If the job is in the RUNNING state, use the cancelJob API to first stop the job, and then use the deleteJob API to delete that job. This API deletes job details and all job results from the database.

URI

```
https://<host>:<port>/ora/managementService/manage/deleteJob
```

HTTP Method

POST

Parameter

jobId— It is an output string. This is a system-generated job ID. See Shared Parameters, on page 112.

Example

HTTPS POST:

```
https://10.194.118.1:8440/ora/managementService/manage/deleteJob
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
 "requestParameters": { "jobId": "JobId_1" }
 }
```

Response:

```
{ "responseMessage": "Success: Your request was successfully
   completed.", "responseCode": 2000, "jobId": "JobId_1" }
```

CHAPTER 5

Job Query APIs

Introduction

Use the Job query APIs to retrieve the status and results for jobs created using the job management APIs.

The getJobsById API allows you to retrieve existing jobs using the jobId parameter. Subsequent calls to this API may return different values as the job is still in progress.

Within the getJobs API, you can specify specific parameters to obtain results based on your search criteria.

If you have not provided a valid job state, MediaSense may not be able to process your request. In this case, verify the job state and issue the request again. All job states are listed in Job States, on page 20.

getJobById

Use this API to retrieve existing jobs using the jobId. In successful JSON response fields, the jobstate, operationsCompleted, and operationsRemaining parameter values are retrieved at the time the API is called. Subsequent calls to this API may display different values.

URI

```
https://<host>:<port>/ora/queryService/query/getJobById
```

HTTP Method

GET

Parameters

- jobDuration— It is an output integer for the API. See Shared Parameters, on page 112.

- jobId— It is an output string. It is a system-generated Id. See Shared Parameters, on page 112.
- jobState— It is an output string for the API. See Shared Parameters, on page 112.
- jobs— It is an output array of job objects. See Shared Parameters, on page 112.
- jobStartTime— It is an output integer for the API. See Shared Parameters, on page 112.
- jobType— It is an output string for the API. See Shared Parameters, on page 112.
- operationsCompleted— It is an output integer. It is the number of operations completed for this job. The number varies depending on when the parameter is called. The counter starts at zero (0).
- operationsRemaining— It is an output integer. The number of operations remaining for this job. The number varies depending on when the parameter is called.

Example

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getJobById?jobId=AMS_10.194.118.1_1282948026491_5
```

Headers:

JSESSIONID: <the jsessionId received from a previous signIn request>

Response:

```
{
    "responseMessage": "Success: Your request was
     successfully completed.",
    "responseCode": 2000,
    "responseBody": {
        "jobs": [
            {
                "jobType": "BULK_DELETE_SESSIONS",
                "jobDuration": 203567,
                "jobStartTime": 1343333766345,
                "jobId": "AMS_10.194.118.1_
                 1282948026491_5",
                "jobState": "COMPLETED"
            }
        ],
        "operationsCompleted": 2,
        "operationsRemaining": 0
    }
}
```

getJobResult

Use this API to retrieve job results of existing jobs. The "requestParameters" parameter is optional. If it is not specified, all job operations for the specified job Id will be returned. If the query filter specified in the "requestParameters" parameter does not match any job operation for the specified job Id, then the response will not contain the "jobOperationResultset" parameter.

URI

```
https://<host>:<port>/ora/queryService/query/getJobResult
```

HTTP Method

POST

Parameters

- byFieldName— It is an optional input string. Enumerations allowed for this parameter are:
    - operationId
    - operationData
    - operationResponse

  See Shared Parameters, on page 112.
- fieldConditions— It is a required input. See Shared Parameters, on page 112.
- fieldConnector— It is an optional string when you have one field in an array. It is a required input string when you have two or more fields in an array. See Shared Parameters, on page 112.
- fieldName— It is a required input string. See Shared Parameters, on page 112.
- fieldOperator— It is a required input string. See Shared Parameters, on page 112.
- fieldValues— It is a required input. See Shared Parameters, on page 112.
- jobDuration— It is an output integer. See Shared Parameters, on page 112.
- jobId— It is an output string. It is a system-generated Id. See Shared Parameters, on page 112.
- jobOperationsResultset— It is output array of strings. It represents set of individual operations results for a particular job.
- jobState— It is an output string for the API. See Shared Parameters, on page 112.
- jobStartTime— It is an output integer for the API. See Shared Parameters, on page 112.
- jobType— It is an output string for the API. See Shared Parameters, on page 112.
- limit— It is a required input integer. See Shared Parameters, on page 112.
- order— It is an optional input string. See Shared Parameters, on page 112.
- operationData— It is an output string and an optional input string. It is case sensitive. It is the individual operation input data in a job.
- operationId— It is an output string and an optional input string. It is case sensitive. It is the individual operation in a job.
- operationResponse— It is an output string and an optional input string. It is case sensitive. It is the individual operation in a job.
- pageParameters— It is an optional input. See Shared Parameters, on page 112.
- paramConnector— It is an optional string when you have one field in an array. It is a required input string when you have two or more fields in an array. See Shared Parameters, on page 112.
- sortParameters— It is an optional input JSON array. See Shared Parameters, on page 112.

fieldOperators for getJobResult

Parameter	Allowed fieldOperators for this parameter
operationId	equals
operationData	equals
operationResult	equals
operationFailureReason	equals contains startsWith endsWith

Example

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/
getJobResult
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn
request>
```

Body:

```
{
    "jobId": "AMS_10.27.185.20_1287093966514_162",
    "requestParameters": [
        {
            "fieldName" : "operationResult",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : [
                        "2000"
                    ]
                }
            ]
        }
    ],
    "sortParameters": [
        {
            "byFieldName": "operationResult",
            "order": "ASC"
        }
    ]
}
```

Response:

```
{
    "responseMessage": "Success: Your request was
     successfully completed.",
    "responseCode": 2000,
    "responseBody": {
        "job": {
            "jobType": "BULK_DELETE_SESSIONS",
            "jobDuration": 36005,
            "jobId": "AMS_10.27.185.20_1287093966514_
             162",
            "jobStartTime": 1287171904409,
            "jobState": "COMPLETED"
        },
        "jobOperationsResultset": [
             {
                "responseCode": 2000,
                "operationData": "Session-1-10.194.118.
                 92-1287017042781",
                "jobOperationId": "AMS_10.27.185.20_
                 1287093966514_163",
                "OperationResponse": "Successful: Your
                 request was successfully completed."
            },

            {
                "responseCode": 2000,
                "operationData": "Session-11-10.194.118.
                 92-1287178816956",
                "jobOperationId": "AMS_10.27.185.20_
                 1287093966514_164",
                "OperationResponse": "Successful: Your
                 request was successfully completed."
            }
        ]
    }
}
```

getJobs

Use this API to retrieve existing jobs using any of the job parameter names.

URI

`https://<host>:<port>/ora/queryService/query/getJobs`

HTTP Method

POST

Parameters

- byFieldName— It is an optional input string. See Shared Parameters, on page 112.

  Allowed enumerations for this parameter.

  - jobId
  - jobState
  - jobType
  - jobStartTime
  - jobDuration

- fieldConditions— It is a required array of strings. See Shared Parameters, on page 112.
- fieldConnector— It is an optional input string when you have one field in an array. It is a required input string when you have two or more fields in an array. See Shared Parameters, on page 112.
- fieldName— It is a required input string. See Shared Parameters, on page 112.

- fieldOperator— It is a required input string. See Shared Parameters, on page 112.

fieldOperators for getJobs

Parameter	Allowed fieldOperators for the Parameter
jobId	equals
jobState	equals
jobType	equals
jobStartTime	between lessThan greaterThan
jobDuration	between lessThan greaterThan

- fieldValues— It is a required array of strings. See Shared Parameters, on page 112.
- jobDuration— It is an optional input integer. See Shared Parameters, on page 112.
- jobId— It is an optional input string. See Shared Parameters, on page 112.
- jobState— It is an optional input string. See Shared Parameters, on page 112.
- jobs— It is an output array of objects. See Shared Parameters, on page 112.
- jobStartTime— It is an optional input integer. See Shared Parameters, on page 112.
- jobType— It is an optional input string. See Shared Parameters, on page 112.
- limit— It is a required input integer. See Shared Parameters, on page 112.
- order— It is an optional input string. However, it becomes a required input string within sortParameters. See Shared Parameters, on page 112.
- pageParameters— It is an optional input JSON object. See Shared Parameters, on page 112.
- paramConnector— It is an optional input string when you have one field in an array. It is a required input string when you have two or more fields in an array. See Shared Parameters, on page 112.
- sortParameters— It is an optional input JSON array. See Shared Parameters, on page 112.

Example

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/
getJobs
```

Headers:

```
Content-Type: application/json
```

```
JSESSIONID: <the jsessionId received from a
signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "jobState",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : [
                        "completed"
                    ]
                }
            ]
        }
    ],
    "sortParameters": [
        {
            "byFieldName": "jobId",
            "order": "ASC"
        }
    ]
}
```

Response:

```
{
    "responseMessage": "Success: Your request was
     successfully completed.",
    "responseCode": 2000,
    "responseBody": {
        "jobs": [
            {
                "jobType": "BULK_DELETE_SESSIONS",
                "jobDuration": 34200,
                "jobStartTime": 1343334076111,
                "jobId": "Job_123",
                "jobState": "COMPLETED"
            },
            {
                "jobType": "BULK_DELETE_SESSIONS",
                "jobDuration": 44200,
                "jobStartTime":1343334075432,
                "jobId": "Job_234",
                "jobState": "COMPLETED"
            },
            {
                "jobType": "BULK_DELETE_SESSIONS",
                "jobDuration": 43678,
                "jobStartTime": 1343334073567,
                "jobId": "Job_345",
                "jobState": "COMPLETED"
            }
        ]
    }
}
```

CHAPTER 6

Recording Control APIs

Introduction

The recording control APIs provided by MediaSense enable a third-party client to control recording of sessions.

Third-party clients can pause or resume a session recording that is currently in progress (active).The following conditions apply to these APIs:

- You can only use this API on active session recordings.
- You can only resume a paused recording.

MediaSense also provides another form of control by allowing third-party clients to trigger a call from MediaSense to a phone, and then start a recording as soon as the call is answered. This is known as Direct Outbound Recording. A corresponding API can stop a recording that has been started in this manner. You can use these APIs to record blogs or video messages.

Note Direct Outbound Recording is supported only for Unified Communications Manager phones.

A call initiated by the startRecording API on a device reference must be answered for the recording to start. You can stop the recording either by hanging up the device or by invoking the stopRecording API on that Device Reference.

Mid-call codec changes are not supported for direct inbound or direct outbound calls.

pauseRecording

Use this API to pause the recording of a session.

URI

```
https://<host>:<port>/ora/controlService/control/pauseRecording
```

HTTP Method

POST

Parameter

sessionId— It is a system-generated identifier for a session. See Shared Parameters, on page 112.

Related Event

tagEvent— The event is sent when a tag is added or deleted from a session. For more information, see tagEvent, on page 135.

Example

To pause a recording that is currently in progress:

HTTPS POST:

```
https://10.194.118.1:8440/ora/controlService/control/
pauseRecording
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   requestParameters: {
                        "sessionId": "Session-1234.abc.
                         5678"
                      }
}
```

Response:

```
 {
    "responseCode": 2000,
    "responseMessage": "Successful"
}
```

resumeRecording

Use this API to resume recording a session.

URI

`https://<host>:<port>/ora/controlService/control/resumeRecording`

HTTP Method

POST

Parameter

sessionId— It is a system-generated identifier for a session. See Shared Parameters, on page 112.

Related Event

tagEvent— The event is sent when a tag is added or deleted from a session. For more information, see tagEvent, on page 135.

Example

HTTPS POST:

```
 https://10.194.118.1:8440/ora/controlService/control/
resumeRecording
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn
request>
```

Body:

```
{
   requestParameters: {
                        "sessionId": "Session-1234.abc.
                         5678"
                      }
}
```

Response:

```
{
    "responseCode": 2000,
    "responseMessage": "Successful"
}
```

startRecording

Use this API to have MediaSense call a phone and record the recipient's audio and video input.

URI

`https://<host>:<port>/ora/controlService/control/startRecording`

HTTP Method

POST

Parameters

- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl— It is an output string. The HTTPS link for a session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. It indicates whether the participant is a conference bridge or an individual device. See Shared Parameters, on page 112.
- mediaType— It is an output string. The type of the media being established. See Shared Parameters, on page 112.
- mediaStreams— It is a required input array of media types. The media stream being established. Each stream is a media type. Therefore, mediaStreams is an array of media types. A recording must have at least one media stream. The array is not ordered. It is a required input.
- mp4Url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- offset— It is an optional input integer. It becomes a required input integer within pageParameters. The first record to be returned. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when this track's recording started. See Shared Parameters, on page 112.
- rtspUrl— It is an output string. The reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionId— It is a required input string. The system-generated unique identifier of a session. See Shared Parameters, on page 112.
- sessionStartDate— It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when the session recording started. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112.
- tags— It is an output JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when the tag was created. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording, which is not case sensitive. See Shared Parameters, on page 112.
- tagOffset— It is an output integer. The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.

- tagType— It is an output string. See Shared Parameters, on page 112.
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. The system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when the track recording started. See Shared Parameters, on page 112.
- urls— It is an output JSON object. It includes information on the httpUrl and the rtspUrl parameters. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.
- xRefCi— It is an output string. The Unified Communications Manager identifier for a particular media stream. See Shared Parameters, on page 112.

Note This API requires that you configure a Call Control Service Provider in Cisco MediaSense Administration, and that the target phone be accessible by that Unified Communications Manager.

Related Event

sessionEvent— It is an event that is sent each time a recording session is started or an existing recording session is updated or ended. For more information, see sessionEvent, on page 131.

Example

Example 1

To start a blog recording that contains both audio and video tracks on a device with deviceRef 1000:

HTTPS POST:

```
https://10.194.118.1:8440/ora/controlService/control/
startRecording
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn
request>
```

Body:

```
{
    "requestParameters": {
        "deviceRef": "1000",
        "mediaStreams": [
            {
                "mediaType": "VIDEO"
            },
            {
                "mediaType": "AUDIO"
            }
        ]
    }
}
```

Example 2

To start a blog recording that contains only an audio track on a device with deviceRef 1111:

HTTPS POST:

```
https://10.194.118.1:8440/ora/controlService/control/
startRecording
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn
request>
```

Body:

```
{
    "requestParameters": {
        "deviceRef": "1111",
        "mediaStreams": [
            {
                "mediaType": "AUDIO"
            }
        ]
    }
}
```

stopRecording

Use this API to stop device recording as soon as an outbound call (initiated by the startRecording API) is completed.

URI

`https://<host>:<port>/ora/controlService/control/stopRecording`

HTTP Method

POST

Parameters

- codec— The codec of the track. See Shared Parameters, on page 112.
- deviceId— The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl— The HTTPS link for a session. See Shared Parameters, on page 112.
- isConference— It indicates whether the participant is a conference bridge or an individual device. See Shared Parameters, on page 112.
- mp4Url— The mp4 link for the session. See Shared Parameters, on page 112.

- offset— The first record to be returned. See Shared Parameters, on page 112.
- participantDuration— The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is a JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— The number of milliseconds since Jan 1, 1970 GMT when this track's recording started. See Shared Parameters, on page 112.
- rtspUrl— The reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— The JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration— The number of milliseconds that the session lasted. See Shared Parameters, on page 112.
- sessionId— The system-generated unique identifier of a session. See Shared Parameters, on page 112.
- sessionStartDate— The number of milliseconds since Jan 1, 1970 GMT when the session recording started. See Shared Parameters, on page 112.
- sessionState— The state of the session. See Shared Parameters, on page 112.
- tags— The JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— The number of milliseconds since Jan 1, 1970 GMT when the tag was created. See Shared Parameters, on page 112.
- tagName— The name that is used to label a recording, which is not case sensitive. See Shared Parameters, on page 112.
- tagOffset— The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112.
- trackDuration— The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— The system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— The JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— The number of milliseconds since Jan 1, 1970 GMT when the track recording started. See Shared Parameters, on page 112.
- urls— It includes information on the httpUrl and the rtspUrl parameters. See Shared Parameters, on page 112.
- wavUrl— The wav link for the session. See Shared Parameters, on page 112.
- xRefCi— The Unified Communications Manager identifier for a particular media stream. See Shared Parameters, on page 112.

Related Event

sessionEvent— It is an event that is sent each time a recording session is started or an existing recording session is updated or ended. For more information, see sessionEvent, on page 131.

Example

To stop a blog recording in progress on a device with deviceRef 1000:

HTTPS POST:

```
https://10.194.118.1:8440/ora/controlService/control/
stopRecording
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{ "requestParameters":
   {
                            "deviceRef": "1000"
    }
}
```

launchMediaPlayer

Use this API to launch the MediaSense media player and play the specified recording. The API is designed to work with a server-based application that is acting as a proxy for a browser. However, a browser can directly call the API. When an application issues this request on behalf of a browser client, it passes the JNLP file to that browser for execution. The browser then interacts with MediaSense directly to download the player application and to fetch the media stream to be played. The API parameters are JSON and not URL.

URI

`https://<host>:<port>/ora/controlService/control/launchMediaPlayer`

HTTP Method

GET

Parameter

rtspUrl— The parameter must be specified in the query section of the URL:

rtspUrl=<user provided string>

where rtspUrl is the rtsp URL of the session to be played in the media player when it launches. This URL is obtained with the Query APIs (such as getSessions). The client fetches this rtspUrl from MediaSense shortly before using it in the launchMediaPlayer request. As with all URLs provided by MediaSense, the rtspUrl for a given recorded session is subject to change from time to time without notice.

Note You must have the required version of Java installed on the client machine. See the Search and Play section of the MediaSense User Guide for more details on Java requirements, available at http://www.cisco.com/c/en/us/support/customer-collaboration/mediasense/tsd-products-support-maintain-and-operate.html.

Example

Example 1

To play the session with the rtspUrl "13092154564622" in the Media Player:

HTTPS GET:

https://<server>:<port>/ora/controlService/control/launchMediaPlayer?rtspUrl=rtsp://<server>/archive/2413e6746c8441

Headers: JSESSIONID (the jsessionID received from a previous signIn request)

Response:

Note This response is for illustrative purposes only. Your application does not need to parse this response. If Java is installed correctly, this response automatically launches the Media Player applet using JWS.

```
...
Content-type: application/x-java-jnlp-file
...

<?xml version="1.0" encoding="UTF-8"?>
<jnlp codebase="https://<server>:<port>/mediasense/
 " spec="1.0+">
   <information>
      <title>MediaSensePlayer</title>
      <vendor>Cisco Systems</vendor>
      <homepage href="http://www.cisco.com" />
      <description>MediaSensePlayer</description>
      <description kind="short">MediaSensePlayer</description>
   </information>
   <update check="always" />
   <security>
      <all-permissions />
   </security>
   <resources>
      <j2se version="1.7.0_11" />
      <jar href="java/MediaSensePlayer.jar" main="true" />
      <jar href="java/jna-3.4.0.jar" />
   </resources>
   <application-desc main-class="com.cisco.cbabu.videoplayer.
    Main">
      <argument>rtsp://<server>/archive/2413e6746c8441?
       token=HQFmZ0yl2ynton0CvOKMifXwe6gBONLAOeMqaPzw2jTrrbhh
       SjbnjOr2siZJmIIqcg2gJDNsFpMrDheQyUGYveUhDQAwdT6ze6mQeek
       JPhipAhtI15Rrfaackd6r6jlj</argument>
   </application-desc>
</jnlp>
```

Example 2

An example of a failure response. (The GET request was sent without the required rtspUrl parameter.)

HTTPS GET: https://<server>:<port>/ora/controlService/control/launchMediaPlayer

Headers: JSESSIONID (the jsessionID received from a previous signIn request)

Response:

```
...
Content-type: application/json
...

{
    "responseMessage": "Failure: Missing parameter in message.",
    "responseCode": 4061,
    "detail": "rtspUrl"
}
```

CHAPTER 7

Session Management APIs

Introduction

In the Contact Center or in speech analytics environments, a supervisor, an agent or a speech analytics system may need to add a tag to a session that is being recorded or has already been recorded. Or, they may need to better manage their session recording repository by deleting some sessions or saving other sessions to prevent those sessions from being deleted.

MediaSense provides these management capabilities through session management APIs. Using these APIs, third-party clients can add or delete tags to sessions and delete sessions one-at-a-time or in bulk. Clients can also convert a session to supported formats like mp4 or wav and move them to a pre-specified location (on the MediaSense system).

Note While you can add or delete tags on both active and completed session recordings, you can only delete completed session recordings. Similarly, you cannot convert a session if it is in the ACTIVE or ERROR state.

addSessionTag

Use this API to add tags to closed (already recorded) or active (currently being recorded) sessions. A tag is the name assigned by the user to label a recording.

Note Within Cisco, the accepted protocol when adding tags is to include a prefix and a colon (:) at the beginning of each tag to identify the application that is inserting the tag. For example, the prefix for tags inserted by CCX applications is "CCX:". Following this protocol ensures that multiple, independent client applications can insert tags into MediaSense without inadvertently inserting tag names that are meaningful or confusing to each other.

URI

```
https://<host>:<port>/ora/managementService/manage/addSessionTag
```

Parameter

- sessionId— It is a required input string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording, which is not case sensitive. See Shared Parameters, on page 112.
- tagOffset— It is an optional integer. The number of milliseconds from the start of session for the tag. See Shared Parameters, on page 112.

Related Event

tagEvent— The event is sent when a tag is added or deleted from a session. For more information, see tagEvent, on page 135.

Example

HTTPS POST:

```
https://10.10.10.10:8440/ora/managementService/manage/
addSessionTag
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   "requestParameters":
                {
                    "sessionId": "AMS_10.194.118.56_1283550575777_2",
                    "tagName": "Sample tag",
                    "tagOffset": 10
               }
}
```

Response:

```
{
     "responseMessage":"Success: Your request was successfully completed.",
     "responseCode":2000
}
```

convertSession (Deprecated)

Use this API to convert a MediaSense session to the format specified in the request. If your request is successful, you receive a list of links of the converted files in the response.

Note

If the MediaSense session exists in the request format, the existing links are returned. New links are not created in the system. If the MediaSense session is in ACTIVE or ERROR state, the session cannot be converted.

The system deletes the converted mp4 and wav files after a maximum of 2 hours of their creation or last modification.

URI

`https://<host>:<port>/ora/managementService/manage/convertSession`

HTTP Method

POST

Parameters

- convertedLink— It is an output string. These strings are generated as part of the convertSession API. A comma-separated list of URLs. Each URL points to the audio recording of a session. All values together make up the string.
- convertedFormat— It is an intput string. The format to convert a MediaSense session. Enumeration value is MP4.
- sessionId— It is a required input string. The system-generated identifier for a session. See Shared Parameters, on page 112.

Examples

Example 1

HTTPS POST:

```
https://10.10.10.10:8440/ora/managementService/manage/
convertSession
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{ "requestParameters":
     {
         "sessionId": "AMS_10.10.10.10_1283550575777_2",
         "conversionFormat": "mp4"
     }
}
```

Response:

```
{
    "responseMessage":"Success: Your request was successfully completed.",
    "responseCode":2000,
    "responseBody":
        {
            "convertedLink":"http://10.10.10.10:8080/
             oramedia/mp4/Session-1-10.10.10.10-
             1283811989534.mp4"
        }
}
```

Example 2—System Capacity Exceeded

HTTPS POST:

```
 https://10.10.10.10:8440/ora/managementService/manage/
convertSession
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{ "requestParameters":
     {
         "sessionId": "AMS_10.10.10.10_1283550575777_2",
         "conversionFormat": "mp4"
     }
}
```

Response:

```
{
    “responseCode”: 5006,
    “responseMessage”: “Failure: Unable to convert the session. Try again later.”,
    "detail": "Capacity Exceeded"
}
```

deleteSessions

Use this API to delete recorded sessions based on the specified Session IDs.

Note You cannot delete an active session.

The jobId parameter in a successful response is used in the job query API to retrieve the job status and results.

URI

`https://<host>:<port>/ora/managementService/manage/deleteSessions`

HTTP Method

POST

Parameters

- jobId— It is an output string. It is a system-generated ID. See Shared Parameters, on page 112.
- sessionIds— It is an input parameter. A list of sessions IDs.

Example

To delete multiple recordings:

HTTPS POST:

```
https://10.194.118.64:8440/ora/managementService/manage/
deleteSessions
```

Headers:

```
Content-Type: application/json
```

Body:

```
{
    "requestParameters": {
        "sessionIds": [
            "4ed31387d58902d1",
            "4ed31387d58923a3"
        ]
    }
}
```

Response:

```
 {
    "responseCode": 2000,
    "responseMessage": "Successful",
    "jobId": "abcd1234"
}
```

deleteSessionTag

Use this API to delete tags from a session. A tag is the name assigned by the user to label a recording. After you no longer need a label or tag, delete it from this session using this API. The tagName and tagOffset parameters are unique for a tag. When deleting a recording, make sure that you use the same name and offset fields with which it was created (using the addSessionTag API). After you delete a tag, you cannot undo this operation. However, you can add the tag again to the session by specifying the tag name and offset fields.

URI

```
https://<host>:<port>/ora/managementService/manage/deleteSessionTag
```

HTTP Method

POST

Parameter

- sessionId— It is an output string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset— It is an output integer. The offset is calculated from the start of the session or track. See Shared Parameters, on page 112.

Related Event

tagEvent— The event is sent when a tag is added or deleted from a session. For more information, see tagEvent, on page 135.

Example

HTTPS POST:

```
https://10.78.95.207:8440/ora/managementService/manage/
deleteSessionTag
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
   "requestParameters":
                {
                  "sessionId": "3212d734fd0a71",
                  "tagName": "demo",
                  "tagOffset": 10
               }
}
```

Response:

```
{
     "responseMessage":"Success: Your request was
      successfully completed.",
     "responseCode":2000
}
```

CHAPTER 8

Session Query APIs

Introduction

MediaSense query APIs allow applications to query for the sessions in flexible ways.

- A complex, but flexible getSessions API allows you to perform complex queries. Using this API, you can specify various logical operations and other sort and pagination parameters in the search criteria.
- Simpler, but less flexible query APIs only allow some basic, but commonly-used, queries (for example, getAllActiveSessions).

getAllActiveSessions

Use this API to search all active recordings. The returned sessions are automatically sorted by sessionStartDate (in descending order).

Note The response for this API depends on the query—All parameters may not be available in each response. For example, active sessions do not have a sessionDuration, downloadUrl, and other parameters. Similarly, the httpUrl is only available if the session is converted.

URI

```
https://<host>:<port>/ora/queryService/query/getAllActiveSessions
```

HTTP Method

GET

Parameters

- callControllerIP— It is an output string. See Shared Parameters, on page 112.
- callControllerType— It is an output string. See Shared Parameters, on page 112.
- ccid— It is an output string that is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- errorCode—
- errorDetail— It is an output string. See Shared Parameters, on page 112.
- httpUrl— It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. It indicates whether the participant is a conference bridge or an individual device. See Shared Parameters, on page 112.
- limit— It is a required input integer. The number of records to be returned, starting at the specified offset. See Shared Parameters, on page 112.
- maxSessionStartDate— It is an optional input integer. The number of milliseconds since Jan 1, 1970 GMT when the session recording started, or more precisely, when the first track for this session began recording. See Shared Parameters, on page 112.

- minSessionStartDate— It is an optional input integer. The number of milliseconds since Jan 1, 1970 GMT when the session recording started, or more precisely, when the first track for this session began recording. See Shared Parameters, on page 112.
- mp4Url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- offset— It is an optional input integer. The first record to be returned. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when this track's recording started. See Shared Parameters, on page 112.
- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration— It is an output integer. The number of milliseconds that the session lasted. See Shared Parameters, on page 112.
- sessionId— It is an output string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- sessionStartDate— It is an output integer. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112.
- tags— It is an ouput JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when this tag was created. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset— It is an output integer. The offset is calculated from the start of the session or track. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112.
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. The system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when the track recording started.
- urls— It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.

- xRefCi— It is an output string. The Unified Communications Manager identifier for a particular media stream. See Shared Parameters, on page 112.

Examples

Example 1

To get all active sessions in the last 2 hours ending at 26 Jul 2012 20:45:40 GMT (for example, from 18:45:40 to 20:45:40):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/
query/getAllActiveSessions?maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous
signIn request>
```

Example 2

To get the first ten active sessions between 26 Jul 2012 18:45:40 GMT and 26 Jul 2012 20:45:40 GMT (2-hour time window):

HTTPS GET:

```
https://10.194.118.1:8443/ora/queryService/query/
getAllActiveSessions?offset=0&limit=10&
minSessionStartDate=1343328340000&maxSessionStartDate
=1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous
signIn request>
```

getAllPrunedSessions

Use this API to search all pruned recordings. The returned sessions are automatically sorted by sessionStart date. The term Pruned refers to recordings that are deleted by the MediaSense system. If you have explicitly deleted any recording using the deleteSessions API, then these deleted recordings, not pruned recordings.

URI

```
https://<host>:<port>/ora/queryService/query/getAllPrunedSessions
```

HTTP Method

GET

Parameters

- callControllerIP: It is an output string. See Shared Parameters, on page 112.
- callControllerType: It is an output string. See Shared Parameters, on page 112.

- ccid: It is an output string that is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112.
- codec: It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId: It is an output string. The unique identifier of the device. See Shared Parameters.
- deviceRef: It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl: It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl: It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference: It is an output boolean. It indicates whether the participant is a conference bridge or an individual device. See Shared Parameters, on page 112.
- limit: It is a required input integer. The number of records to be returned, starting at the specified offset. See Shared Parameters, on page 112.
- maxSessionStartDate: It is an optional input integer. The number of milliseconds since Jan 1, 1970 GMT when the session recording started, or more precisely, when the first track for this session began recording. See Shared Parameters, on page 112.
- minSessionStartDate: It is an optional input integer. The number of milliseconds since Jan 1, 1970 GMT when the session recording started, or more precisely, when the first track for this session began recording. See Shared Parameters, on page 112.
- mp4Url: It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- offset: It is an optional input integer. The first record to be returned. See Shared Parameters, on page 112.
- participantDuration: It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants: It is output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate: It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when this track's recording started. See Shared Parameters, on page 112.
- rtspUrl: It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions: It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration: It is an output integer. The number of milliseconds that the session lasted. See Shared Parameters, on page 112.
- sessionId: It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- sessionStartDate: It is an output integer. See Shared Parameters, on page 112.
- sessionState: It is an output string. The state of the session. See Shared Parameters, on page 112.
- tags: It is an ouput JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate: It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when this tag was created. See Shared Parameters, on page 112.

- tagName: It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset: It is an output integer. The offset is calculated from the start of the session or track. See Shared Parameters, on page 112.
- tagType: It is an output string. See Shared Parameters, on page 112.
- trackDuration: It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType: It is an output integer. See Shared Parameters, on page 112.
- trackNumber: It is an output integer. The system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks: It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate: It is an output integer. The number of milliseconds since Jan 1, 1970 GMT when the track recording started. See Shared Parameters, on page 112.
- urls: It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl: It is an output string. The wav link for the session. See Shared Parameters, on page 112.
- xRefCi: It is an output string. The Unified Communications Manager identifier for a particular media stream. See Shared Parameters, on page 112.

Examples

Example 1

To get all the pruned sessions in the last 2 hours ending at 26 Jul 2012 20:45:40 GMT (for example, from 18:45:40 to 20:45:40):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getAllPrunedSessions?maxSessionStartDate=1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous
signIn request>
```

Example 2

To get the first ten pruned sessions between 26 Jul 2012 18:45:40 GMT and 26 Jul 2012 20:45:40 GMT (2 hour time window):

HTTPS GET:

```
 https://10.194.118.1:8440/ora/queryService/query/
getAllPrunedSessions?offset=0&limit=10&
minSessionStartDate=1343328340000&maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous
signIn request>
```

getArchiveSessions

Use this API to search and retrieve archived sessions.

URI

```
https://<host>:<port>/ora/queryService/query/getArchiveSessions
```

HTTP Method

POST

Parameters

- deviceRef: It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- sessionIdList: The parameter sessionIdList accepts a list of multiple comma-separated sessionIds in the request body, but currently, only the first sessionId will be considered to return the strongly associated sessionIds. Additional sessionIds will be ignored. This input parameter has been made of the *List* type for future enhancement.
- maxSessionStartDate: It is an optional input integer. However, it becomes a required input integer when no value is specific for minSessionStartDate. See Shared Parameters, on page 112.
- minSessionStartDate: It is an optional input integer. See Shared Parameters, on page 112.

Note Consider the following points while searching for an archived session.

- If you have both sessionId and deviceRef as request parameters, then getArchiveSessions API considers only the sessionId.
- If you enter more than one sessionId, then getArchiveSessions API considers only the first sessionId.
- If you enter more than one deviceRef, then getArchiveSessions API considers only the first deviceRef.

getAssociatedSessions

Use this API to search and retrieve strongly associated sessions by a sessionID. The strongly associated sessions contain one xRefci in common (for built-in-bridge (BiB) recordings and Unified Communications Manager enabled network-based recordings (NBR)), and at least one CCID in common (for recordings through Unified Border Element).

MediaSense supports call association for sessions generated through the following modes.

- Unified Communications Manager BiB forking
- Unified Communications Manager NBR
- Unified Border Element Dial Peer

Note The Call Association feature works by iterating through the stored recordings and searching for common xRefCi values (for recordings through Unified Communications Manager), and for common CCID values (for recordings through Unified Border Element).

For example:

1. If a conversation being recorded involves a call transfer from one agent (with BiB activated) to another agent (with BiB activated), the recording sessions generated will be linked to each other through a chain of common xRefCi values.
2. If a conversation being recorded through Unified Border Element involves a mid-call codec change, such as A called B and call is negotiated at codec X. A puts the call on hold where music-on-hold (MoH) codec is Y. So, there is another new session for MoH. When A resumes the call, another third session is generated. These generated recording sessions are linked to each other by means of common CCID values.

The Call Association feature takes advantage of this and prepares a set of recording sessions which are thus related to each other.

Note Some call scenarios result in isolated recording sessions (sessions none of whose xRefCi or CCID are common with any other session in the database). Such sessions cannot be associated through this feature.

URI

```
https://<host>:<port>/ora/queryService/query/getAssociatedSessions
```

HTTP Method

POST

Parameters

- callControllerType— It is an output string. See Shared Parameters, on page 112.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- firstName— It is an output string. See Shared Parameters, on page 112.
- httpUrl— It is an output string. See Shared Parameters, on page 112.
- isConference— It is an output boolean. It indicates whether the participant is a conference bridge or an individual device. See Shared Parameters, on page 112.
- lastName— It is an output string. See Shared Parameters, on page 112.

- lineDisplayName— It is an output string. See Shared Parameters, on page 112.
- loginId— It is an output string. See Shared Parameters, on page 112.
- loginIdDomain— It is an output string. See Shared Parameters, on page 112.
- loginName— It is an output string. See Shared Parameters, on page 112.
- mp4Url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participantInformation— It is an output JSON array of participant information. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. See Shared Parameters, on page 112.
- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks.
- sessionId— It is an output string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- sessionIdList— It accepts a list of multiple comma-separated sessionIds in the request body, but currently, only the first sessionId will be considered to return the strongly associated sessionIds. Additional sessionIds will be ignored. This input parameter has been made of the *List* type for future enhancement.
- sessionStartDate— It is an output integer. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112.
- trackDuration— It is an output integer. The number of milliseconds the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. The system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. See Shared Parameters, on page 112.
- urls— It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.
- xRefCi— It is a output string. The Unified Communications Manager identifier for a particular media stream. See Shared Parameters, on page 112.

Example

HTTPS POST:

```
https://10.78.170.114:8440/ora/queryService/query/getAssociatedSessions
```

Headers:

```
Content-Type: application/json
```

Body:

```
{
   "requestParameters":{
        "sessionIdList" : ["514406fbae4a1"]
    }
}
```

Response:

```
{
   "responseMessage":"Success: Your request was successfully completed.",
   "responseCode":2000,
   "responseBody":{
      "sessions":[
         {
            "sessionState":"CLOSED_NORMAL",
            "callControllerType":"Cisco-CUCM",
            "sessionId":"614406fbae5e1",
            "urls":{

"httpUrl":"https://10.78.170.114:19443/recordedMedia/oramedia/mp4/614406fbae5e1.mp4",
               "rtspUrl":"rtsp://10.78.170.114/archive/614406fbae5e1",

"mp4Url":"https://10.78.170.114:19443/recordedMedia/oramedia/mp4/614406fbae5e1.mp4",

"wavUrl":"https://10.78.170.114:19443/recordedMedia/oramedia/wav/614406fbae5e1.wav"
            },
            "sessionStartDate":1391686561,
            "tracks":[
               {
                  "trackStartDate":1391686561,
                  "trackDuration":78119,
                  "codec":"PCMU",

"downloadUrl":"http://10.78.170.114:8081/mma/ExportRaw?recording=614406fbae5e1-TRACK1",
                  "trackNumber":1,
                  "trackMediaType":"AUDIO",
                  "participants":[
                     {
                        "participantStartDate":1391686561,
                        "deviceRef":"1341051",
                        "participantInformation": {
                                "loginId": "davpete",
                                "lastName": "Peter",
                                "firstName": "Dave",
                                "loginIdDomain": 1,
                                "loginName": "Peter"
                             },
                        "lineDisplayName": "Dave Peter",
                        "isConference":false,
                        "xRefCi":"20248680",
                        "participantDuration":78119,
                        "deviceId":"SEP70F39517B88A"
                     }
                  ]
               },
               {
                  "trackStartDate":1391686561,
                  "trackDuration":78119,
                  "codec":"PCMU",

"downloadUrl":"http://10.78.170.114:8081/mma/ExportRaw?recording=614406fbae5e1-TRACK0",
                  "trackNumber":0,
                  "trackMediaType":"AUDIO",
                  "participants":[
                     {
                        "participantStartDate":1391686561,
                        "deviceRef":"1341052",
                        "participantInformation": {
                                "loginId": "davpete",
                                "lastName": "Peter",
                                "firstName": "Dave",
                                "loginIdDomain": 1,
                                "loginName": "Peter"
                            },
                        "lineDisplayName": "Dave Peter",
                        "isConference":false,
                        "xRefCi":"20248679",
                        "participantDuration":78119,
                        "deviceId":"SEP402CF4ECA126"
                    }
                ]
            }
        ],
        "sessionDuration":78119,
        "callControllerIP":"10.65.157.134"
    },
    {
        "sessionState":"CLOSED_NORMAL",
        "callControllerType":"Cisco-CUCM",
        "sessionId":"514406fbae4a1",
        "urls":{

"httpUrl":"https://10.78.170.114:19443/recordedMedia/oramedia/mp4/514406fbae4a1.mp4",
            "rtspUrl":"rtsp://10.78.170.114/archive/514406fbae4a1",

"mp4Url":"https://10.78.170.114:19443/recordedMedia/oramedia/mp4/514406fbae4a1.mp4",

"wavUrl":"https://10.78.170.114:19443/recordedMedia/oramedia/wav/514406fbae4a1.wav"
        },
        "sessionStartDate":1391686561,
        "tracks":[
            {
                "trackStartDate":1391686561,
                "trackDuration":78093,
                "codec":"PCMU",

"downloadUrl":"http://10.78.170.114:8081/mma/ExportRaw?recording=514406fbae4a1-TRACK1",
                "trackNumber":1,
                "trackMediaType":"AUDIO",
                "participants":[
                    {
                        "participantStartDate":1391686561,
                        "deviceRef":"1341052",
                        "participantInformation": {
                                "loginId": "davpete",
                                "lastName": "Peter",
                                "firstName": "Dave",
                                "loginIdDomain": 1,
                                "loginName": "Peter"
                            },
                        "lineDisplayName": "Dave Peter",
                        "isConference":false,
                        "xRefCi":"20248679",
                        "participantDuration":78093,
                        "deviceId":"SEP402CF4ECA126"
                    }
                ]
            },
            {
                "trackStartDate":1391686561,
                "trackDuration":78093,
                "codec":"PCMU",

"downloadUrl":"http://10.78.170.114:8081/mma/ExportRaw?recording=514406fbae4a1-TRACK0",
                "trackNumber":0,
                "trackMediaType":"AUDIO",
                "participants":[
                    {
                        "participantStartDate":1391686561,
                        "deviceRef":"1341051",
                        "participantInformation": {
                                "loginId": "davpete",
                                "lastName": "Peter",
                                "firstName": "Dave",
                                "loginIdDomain": 1,
                                    "loginName": "Peter"
                                },
                        "lineDisplayName": "Dave Peter",
                        "isConference":false,
                        "xRefCi":"20248680",
                        "participantDuration":78093,
                        "deviceId":"SEP70F39517B88A"
                    }
                ]
            }
        ],
        "sessionDuration":78093,
        "callControllerIP":"10.65.157.134"
    }
  ]
 }
}
```

getSessionBySessionId

Use this API to search and retrieve a recorded or live session by its session ID.

URI

`https://<host>:<port>/ora/queryService/query/getSessionBySessionId`

HTTP Method

GET

Parameters

- callControllerIP— It is an output string. See Shared Parameters, on page 112.
- callControllerType— It is an output string. See Shared Parameters, on page 112.
- ccid— It is an output string. It is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- devRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl— It is deprecated in release 10 in favor of mp4url. It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. See Shared Parameters, on page 112.
- mp4url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. See Shared Parameters, on page 112.

- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration— It is an output integer. See Shared Parameters, on page 112.
- sessionId— It is a system-generated unique identifier for a session. See Shared Parameters, on page 112.
- sessionStartDate— It is an output integer. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112.
- tags— It is an output JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset— The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112.
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. It is the system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. See Shared Parameters, on page 112.
- urls— It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.
- xRefCi— It is an output string. The Unified Communications Manager identifier for a particular media stream. See Shared Parameters, on page 112.

Example

To get the session with the sessionId "13092154564622":

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getSessionBySessionId?value=13092154564622
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn
request>
```

getSessions

Use this API to search and retrieve recorded, live, or pruned sessions.

Requirement: To enforce scalability, this API requires a sessionStartDate field condition.

Limitation: The getSessions query syntax is flexible however, is not as general as SQL. Using the getSessions API, you cannot express certain types of complex queries. Specifically, the syntax does not provide an approach to override the built-in precedence rules. Also, it does not offer a method to indicate the order in which the tables are joined and value matching is applied.

Examples:

Example 1

If you have three fields to query: f1, f2, and f3; and if the query is "f1=<query> AND f2=<query> OR f3=<query>", then SQL precedence rules are applied to the fields such that the query translates to: "[(f1=<query> AND f2=<query>) OR (f3=<query>)]."

Due to the use of brackets, the AND operator is evaluated before the OR operator. However, it is not possible to specify that the OR operator gets evaluated before the AND operation because the expression syntax does not provide a method to override the implicit precedence rules.

To summarize the limitation, a query like "[(f1) and (f2 or f3)]" is not possible.

Example 2

For searches which reference *different* fields located within the track object, the AND operator applies to values found in the same track instance, not to values found in different tracks of the same session. For example, a query such as "deviceRef=1000 AND loginName='Smith'" only returns a session in which agent Smith was logged in under extension 1000. It will not return sessions in which agent Smith was *in conversation* with extension 1000, because those two values occur on different tracks.

Moreover, if the *same* field is referenced on both sides of the AND operator, then MediaSense applies them to different tracks within the same session. So a query such as "deviceRef=1000 AND deviceRef=2000" finds sessions in which a user at extension 1000 conversed with another user who was at extension 2000, even though these values are found in different tracks. This special handling does not work when a third track-specific field is added to the query. For example, "deviceRef=1000 AND deviceRef=2000 AND loginName='Smith'" may not produce the expected results because there is a different field name involved. And it is no longer clear whether that value is meant to be found on the same track or on the other track in the same session.

Note

- The sessionStartDate field is no longer bound to contain two field values in case of BETWEEN field operator.
- If only one field value is provided with BETWEEN field operator, then the value is treated as the sessionStartDate_value1 parameter. The sessionStartDate_value2 parameter would be automatically set to the MS server's current time by the getSessions API itself.
- If two field values are provided with BETWEEN field operator, then these values are treated as the sessionStartDate_value1 and sessionStartDate_value2, respectively (same as earlier behavior).
- To enforce scalability, this API requires a sessionStartDate field condition.
- This API no longer supports sessionId as a searchable fieldName.

URI

```
https://<host>:<port>/ora/queryService/query/getSessions
```

HTTP Method

POST

Parameters

- byFieldName— It is an optional input string. See Shared Parameters. The allowed enumerations are:
  - sessionId
  - sessionState
  - sessionDuration
  - sessionStartDate
  - ccid
- callControllerIP— It is an output string. See Shared Parameters, on page 112.
- callControllerType— It is an output string. See Shared Parameters, on page 112.
- ccid— It is an optional input string. It is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112. The allowed fieldOperator is “equals”.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId— It is an optional input string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112. The allowed fieldOperators are:
  - equals
  - contains
  - startsWith

- ◦ endsWith
- ◦ between

- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- fieldConditions— It is a required input array of strings. The conditions specified for the queried field. See Shared Parameters, on page 112.
- fieldConnector— It is used to logically connect a condition with the successive condition. See Shared Parameters, on page 112.
- fieldName— It is a required input string. The name of the queried field. See Shared Parameters.
- fieldOperator— It is a required input string. See Shared Parameters, on page 112.
- fieldValues— It is a required input array of strings. See Shared Parameters, on page 112.
- httpUrl— It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. See Shared Parameters, on page 112.
- limit— It is a required input integer. The number of records to be returned, starting at the specified offset. See Shared Parameters, on page 112.
- mp4Url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- nodeIPAddress— It is an output string. The IP address of the Cisco MediaSense server. See Shared Parameters, on page 112.The allowed fieldOperator is “equals”.
- offset— It is an optional input integer. The first record to be returned. See Shared Parameters, on page 112.
- order— It is an optional input string. See Shared Parameters, on page 112.
- partition— It is an output string. See Shared Parameters, on page 112. The allowed fieldOperator is “equals”. The allowed fieldOperator is “equals”.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. See Shared Parameters, on page 112.
- pruned— It is an output boolean. It indicates whether a session was deleted by the system. The sessions that are deleted by a user are not considered as pruned. Pruned is omitted when a session has not been deleted. Enumeration values are:
    - ◦ TRUE
    - ◦ FALSE

  The allowed fieldOperator is “equals”.
- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.

- sessionDuration— It is an output integer. The number of milliseconds that the session lasted. See Shared Parameters, on page 112. The allowed fieldOperators are:
    - lessThan
    - greaterThan
    - equals
- sessionStartDate— It is a required input integer. See Shared Parameters, on page 112.The allowed fieldOperator is "between".
- sortParameters— It is an optional input JSON array. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112. The allowed fieldOperator is "equals".
- tags— It is an output JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112. The allowed fieldOperators are:
    - equals
    - contains
    - startsWith
    - endsWith
- tagOffset— It is an optional integer for addSessionTag. The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112. The allowed fieldOperator is "equals".
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. It is the system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. See Shared Parameters, on page 112.
- urls— It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.
- xRefCi— It is an output string. It is the Unified Communications Manager identifier for a particular media Stream. See Shared Parameters, on page 112. The allowed fieldOperator is "equals".

Example

Example 1

Query— Get all sessions where ((tagName = "foo" and sessionStartDate is between Nov 2, 2010 21:21:40 & Nov 3, 2010 21:21:40) or (deviceRef = "1000" and sessionState = "active"))

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "tagName",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : ["foo"]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [
                        1288732900000, // Tue, 02 Nov 2010
                        21:21:40 GMT
                        1288819300000  // Wed, 03 Nov 2010
                        21:21:40 GMT
                    ]
                }
            ],
            "paramConnector": "OR"
        },
        {
            "fieldName" : "deviceRef",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : ["1000"]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "sessionState",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : ["active"]
                }
            ]
        }
    ]
}
```

Example 2

Query— Get all sessions where ((xRefCi = 12345678 and tagName = foo) or (sessionState = active and sessionStartDate is between Nov 2, 2010 21:21:40 & Nov 3, 2010 21:21:40))

Note Use care when specifying an "or" connector. The sub-expression before the "or" is not limited by the sessionStartDate in the second subexpression. Ensure that the first subexpression is scalable or you must add a sessionStartDate filter to the first subexpression. In this example, use a filter that specifies "xRefCi." The xrefCi is a scalable field, so there is no need to restrict it by sessionStartDate."tagName" is nonscalable, but because it is connected to the xRefCi filter by the "and" operator, the full subexpression is scalable.

For more information about writing scalable query expressions, see Avoiding Non-Scalable Queries.

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "xRefCi",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : [
                        "12345678"
                    ]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "tagName",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : [
                        "foo"
                    ]
                }
            ],
            "paramConnector": "OR"
        },
        {
            "fieldName" : "sessionState",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : [
                        "active"
                    ]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [
                        1288732900000, // Tue, 02 Nov 2010
                        21:21:40 GMT
                        1288819300000  // Wed, 03 Nov 2010
                        21:21:40 GMT
                    ]
                }
            ]
        }
    ]
}
```

Example 3

Query— Get all sessions where (((deviceRef = "1000" or deviceRef="2000") **or** tagName="foo") **and** sessionState="active")

```
A query where higher precedence is intended for an 'OR'
operation over an 'AND' operation is not possible.
```

Example 4

Query— Get all sessions where (deviceRef = "1000" and sessionStartDate is between Nov 2, 2010 21:21:40 & Nov 3, 2010 21:21:40)

Note Although the simpler getSessionsByDeviceRef API is easier for a simpler query, you may still use getSessions API.

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "deviceRef",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : ["1000"]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [
                        1288732900000, // Tue, 02 Nov 2010
                        21:21:40 GMT
                        1288819300000  // Wed, 03 Nov 2010
                        21:21:40 GMT
                    ]
                }
            ]
        }
    ]
}
```

Example 5

Query— Get all sessions where a track has been recorded on nodeIPAddress "10.35.146.158" and partition "/common" between Nov 2, 2010 21:21:40 & Nov 3, 2010 21:21:40

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "nodeIPAddress",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : ["10.35.146.158"]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "partition",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : ["/common"]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [
                        1288732900000, // Tue, 02 Nov 2010
                        21:21:40 GMT
                        1288819300000  // Wed, 03 Nov 2010
                        21:21:40 GMT
                    ]
                }
            ]
        }
    ]
}
```

Example 6

Query— Get all sessions that are pruned by the system between Nov 2, 2010 21:21:40 & Nov 3, 2010 21:21:40

Query— Get all sessions that are pruned by the system between Nov 2, 2010 21:21:40 & Nov 3, 2010 21:21:40

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "pruned",
            "fieldConditions": [
                {
                    "fieldOperator" : "equals",
                    "fieldValues" : ["true"]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [
                        1288732900000, // Tue, 02 Nov 2010
                        21:21:40 GMT
                        1288819300000  // Wed, 03 Nov 2010
                        21:21:40 GMT
                    ]
                }
            ]
        }
    ]
}
```

Example 7

Query— Get all sessions that lie within a certain date range. For example, between Tue, 02 Nov 2010 21:21:40 GMT & Wed, 03 Nov 2010 21:21:40 GMT

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [
                        1288732900000, // Tue, 02 Nov 2010
                        21:21:40 GMT
                        1288819300000  // Wed, 03 Nov 2010
                        21:21:40 GMT
                    ]
                }
            ]
        }
    ]
}
```

Example 8

Query— Get all sessions where ((deviceRef = 1000 or deviceRef = 2000 or deviceRef = 3000) and sessionStartDate is between Nov 2, 2010 21:21:40 & Nov 3, 2010 21:21:40)

HTTPS POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName": "deviceRef",
            "fieldConditions": [
                {
                    "fieldOperator": "equals",
                    "fieldValues": ["1000"],
                    "fieldConnector": "OR"
                },
                {
                    "fieldOperator": "equals",
                    "fieldValues": ["2000"],
                    "fieldConnector": "OR"
                },
                {
                    "fieldOperator": "equals",
                    "fieldValues": ["3000"]
                }
            ],
            "paramConnector": "AND"
        },
        {
            "fieldName": "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator": "between",
                    "fieldValues": [
                        1288732900000, // Tue, 02 Nov 2010
                        21:21:40 GMT
                        1288819300000 // Wed, 03 Nov 2010
                        21:21:40 GMT
                    ]
                }
            ]
        }
    ]
}
```

Example 9

Query— Get all sessions which started between a given date and the MS server's current time. For example: between 12 Aug 2014 12:00 GMT-07:00 and MS server's current time.

HTTP POST:

```
https://10.194.118.1:8440/ora/queryService/query/getSessions
```

Headers:

```
Content-Type: application/json
JSESSIONID: <the jsessionId received from a signIn request>
```

Body:

```
{
    "requestParameters": [
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [407870000000 // 12 Aug 2014 12:00 GMT-07:00]
                }
            ]
        }
    ]
}
```

If MS server's current time is 19 Aug 2014, 12:05 = 1408475100000, the request body of getSessions API is updated to:

```
{
    "requestParameters": [
        {
            "fieldName" : "sessionStartDate",
            "fieldConditions": [
                {
                    "fieldOperator" : "between",
                    "fieldValues" : [
                        1407870000000, // 12 Aug 2014 12:00 GMT-07:00
                        1408475100000  // 19 Aug 2014 12:05 GMT-07:00
                    ]
                }
            ]
        }
    ]
}
```

getSessionsByCCID

Use this API to search and retrieve recorded or live sessions by Call Correlation Identifier (CCID). The returned sessions are automatically sorted by sessionStartDate, in descending order.

URI

```
https://<host>:<port>/ora/queryService/query/getSessionsByCCID
```

HTTP Method

GET

Parameters

- callControllerIP— It is an output string. See Shared Parameters, on page 112.
- callControllerType— It is an output string. See Shared Parameters, on page 112.
- ccid— It is a required input string. It is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.

- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl— It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. See Shared Parameters, on page 112.
- limit— It is a required input integer. The number of records to be returned, starting at the specified offset. See Shared Parameters, on page 112.
- mp4Url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- offset— It is an optional input integer. The first record to be returned. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. See Shared Parameters, on page 112.
- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration— It is an output integer. The number of milliseconds that the session lasted. See Shared Parameters, on page 112.
- sessionId— It is an output string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- sessionStartDate— It is a required input integer. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112. The allowed fieldOperator is “equals”.
- tags— It is an output JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset— It is an optional integer for addSessionTag. The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112. The allowed fieldOperator is “equals”.
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.

- trackNumber— It is an output integer. It is the system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. See Shared Parameters, on page 112.
- urls— It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.
- xRefCi— It is an output string. It is the Unified Communications Manager identifier for a particular media Stream. See Shared Parameters, on page 112.

Examples

Example 1

To get all sessions that have the ccid "0584071424-0000065536-0000000016-0671746570":

HTTPS GET:

```
https://10.194.118.1:8080/ora/queryService/query/
getSessionsByCCID?value=0584071424-0000065536-0000000016-
0671746570&offset=0&limit=0
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn
request>
```

Example 2

To get the first ten sessions that have the ccid "0584071424-0000065536-0000000016-0671746570":

HTTPS GET:

```
https://10.194.118.1:8080/ora/queryService/query/
getSessionsByCCID?value=13092154564622%4010.194.118.71
&offset=0&limit=10
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn
request>
```

Example 3

To get the first 100 (default number of records returned) sessions that have the ccid "0584071424-0000065536-0000000016-0671746570":

HTTPS GET:

```
https://10.194.118.1:8080/ora/queryService/query/
getSessionsByCCID?value=0584071424-0000065536-0000000016-
0671746570
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn
request>
```

Note In the absence of offset or limit parameters, the request defaults to the first 100 sessions to be returned.

getSessionsByDeviceRef

Use this API to search and retrieve recorded or live sessions by Device Ref (the phone number, IP address, or URI/URL for each device). The returned sessions are automatically sorted by sessionStartDate (in descending order).

URI

```
https://<host>:<port>/ora/queryService/query/getSessionsByDeviceRef
```

HTTP Method

GET

Parameters

- callControllerIP— It is an output string. See Shared Parameters, on page 112.
- callControllerType— It is an output string. See Shared Parameters, on page 112.
- ccid— It is an output string. It is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl— It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. See Shared Parameters, on page 112.
- limit— It is a required input integer. The number of records to be returned, starting at the specified offset. See Shared Parameters, on page 112.
- maxSessionStartDate— It is an optional input integer. See Shared Parameters, on page 112.
- minSessionStartDate— It is an optional input integer. See Shared Parameters, on page 112.
- offset— It is an optional input integer. The first record to be returned. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.

- participantStartDate— It is an output integer. See Shared Parameters, on page 112.
- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration— It is an output integer. The number of milliseconds that the session lasted. See Shared Parameters, on page 112.
- sessionId— It is an output string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- sessionStartDate— It is a required input integer. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112.
- tags— It is an output JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset— It is an optional integer for addSessionTag. The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112.
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. It is the system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. See Shared Parameters, on page 112.
- urls— It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.
- xRefCi— It is an output string. The Unified Communications Manager identifier for a particular media stream. See Shared Parameters, on page 112.

Examples

Example 1

To get all sessions that have the deviceRef "1000" in the last 2 hours ending at 26 Jul 2012 20:45:40 GMT (for example, from 18:45:40 to 20:45:40):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getSessionsByDeviceRef?value=1000&maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn request>
```

Example 2

To get the first ten sessions that have the deviceRef "1000" between 26 Jul 2012 18:45:40 GMT and 26 Jul 2012 20:45:40 GMT (2-hour time window):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getSessionsByDeviceRef?value=1000&offset=0&limit=
10&minSessionStartDate=1343328340000&maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn request>
```

getSessionsByMediaType

Use this API to search and retrieve recorded or live sessions based on the media type (audio, video, audio and video, or screen capture). The returned sessions are automatically sorted by sessionStartDate, with the most recent one being on the top.

URI

https://<host>:<port>/ora/queryService/query/getSessionsByMediaType

HTTP Method

GET

Parameters

- callControllerIP— It is an output string. See Shared Parameters, on page 112.
- callControllerType— It is an output string. See Shared Parameters, on page 112.
- ccid— It is an output string. It is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.
- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl— It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. See Shared Parameters, on page 112.

- limit— It is a required input integer. The number of records to be returned, starting at the specified offset. See Shared Parameters, on page 112.
- maxSessionStartDate— It is an optional input integer. See Shared Parameters, on page 112.
- mediaType— It is an output string and a required input string. See Shared Parameters, on page 112.
- minSessionStartDate— It is an optional input integer. See Shared Parameters, on page 112.
- mp4Url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- offset— It is an optional input integer. The first record to be returned. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. See Shared Parameters, on page 112.
- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration— It is an output integer. The number of milliseconds that the session lasted. See Shared Parameters, on page 112.
- sessionId— It is an output string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- sessionStartDate— It is a required input integer. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112.
- tags— It is an output JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset— It is an optional integer for addSessionTag. The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112.
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.
- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. It is the system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. See Shared Parameters, on page 112.
- urls— It is an output JSON object. See Shared Parameters, on page 112.

- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.

Examples

Example 1

To get all sessions that have the mediaType "AUDIO" in the last 2 hours ending at 26 Jul 2012 20:45:40 GMT (for example, from 18:45:40 to 20:45:40):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getSessionsByMediaType?value=AUDIO&maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn request>
```

Example 2

To get the first ten sessions that have the mediaType "AUDIO" between 26 Jul 2012 18:45:40 GMT and 26 Jul 2012 20:45:40 GMT (2-hour time window):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getSessionsByMediaType?value=AUDIO&offset=0&limit=
10&minSessionStartDate=1343328340000&maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn request>
```

getSessionsByTag

Use this API to search and retrieve recorded or live sessions based on the user-assigned tag names. A tag is the name a user assigns to label a recording. The returned sessions are automatically sorted by sessionStartDate (in descending order).

URI

`https://<host>:<port>/ora/queryService/query/getSessionsByTag`

HTTP Method

GET

Parameters

- callControllerIP— It is an output string. See Shared Parameters, on page 112.
- callControllerType— It is an output string. See Shared Parameters, on page 112.
- ccid— It is an output string. It is used to identify recording sessions which are part of the same call. See Shared Parameters, on page 112.
- codec— It is an output string. The codec of the track. See Shared Parameters, on page 112.

- deviceId— It is an output string. The unique identifier of the device. See Shared Parameters, on page 112.
- deviceRef— It is an output string. The phone number of each device. See Shared Parameters, on page 112.
- downloadUrl— It is an output string. The URL that is used to download the recording in the raw format. See Shared Parameters, on page 112.
- httpUrl— It is an output string. The HTTPS link for the session. See Shared Parameters, on page 112.
- isConference— It is an output boolean. See Shared Parameters, on page 112.
- limit— It is a required input integer. The number of records to be returned, starting at the specified offset. See Shared Parameters, on page 112.
- maxSessionStartDate— It is an optional input integer. See Shared Parameters, on page 112.
- minSessionStartDate— It is an optional input integer. See Shared Parameters, on page 112.
- mp4Url— It is an output string. The mp4 link for the session. See Shared Parameters, on page 112.
- offset— It is an optional input integer. The first record to be returned. See Shared Parameters, on page 112.
- participantDuration— It is an output integer. The number of milliseconds that the participant was active in the session. See Shared Parameters, on page 112.
- participants— It is an output JSON array of participant objects. See Shared Parameters, on page 112.
- participantStartDate— It is an output integer. See Shared Parameters, on page 112.
- rtspUrl— It is an output string. A reference to the entire session, which can contain multiple tracks. See Shared Parameters, on page 112.
- sessions— It is an output JSON array of session objects. See Shared Parameters, on page 112.
- sessionDuration— It is an output integer. The number of milliseconds that the session lasted. See Shared Parameters, on page 112.
- sessionId— It is an output string. It is a system-generated identifier for a session. See Shared Parameters, on page 112.
- sessionStartDate— It is a required input integer. See Shared Parameters, on page 112.
- sessionState— It is an output string. The state of the session. See Shared Parameters, on page 112.
- tags— It is an output JSON array of tag objects. See Shared Parameters, on page 112.
- tagCreateDate— It is an output integer. See Shared Parameters, on page 112.
- tagName— It is an output string. The name that is used to label a recording. See Shared Parameters, on page 112.
- tagOffset— It is an optional integer for addSessionTag. The number of milliseconds from the start of session for this tag. See Shared Parameters, on page 112.
- tagType— It is an output string. See Shared Parameters, on page 112.
- trackDuration— It is an output integer. The number of milliseconds that the track lasted. See Shared Parameters, on page 112.

- trackMediaType— It is an output integer. See Shared Parameters, on page 112.
- trackNumber— It is an output integer. It is the system-generated unique identifier of the track. See Shared Parameters, on page 112.
- tracks— It is an output JSON array of track objects. See Shared Parameters, on page 112.
- trackStartDate— It is an output integer. See Shared Parameters, on page 112.
- urls— It is an output JSON object. See Shared Parameters, on page 112.
- wavUrl— It is an output string. The wav link for the session. See Shared Parameters, on page 112.

Examples

Example 1

To get all sessions that have the tag "mysampletag1" in the last 2 hours ending at 26 Jul 2012 20:45:40 GMT (for example, from 18:45:40 to 20:45:40):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getSessionsByTag?value=mysampletag1&maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous signIn request>
```

Example 2

To get the first ten sessions that have the tag "mysampletag1" between 26 Jul 2012 18:45:40 GMT and 26 Jul 2012 20:45:40 GMT (2-hour time window):

HTTPS GET:

```
https://10.194.118.1:8440/ora/queryService/query/
getSessionsByTag?value=mysampletag1&offset=0&limit=
10&minSessionStartDate=1343328340000&maxSessionStartDate=
1343335540154
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous
signIn request>
```

Concurrent Search Requests

To ensure that system performance is not impacted, MediaSense implements the following rules for session query API requests:

- The MediaSense system accepts 15 incoming session query API requests and wait lists an additional 10 requests. The system rejects all other requests (beyond these 25 requests).
- If the MediaSense system does not complete a session query API request within two (2) minutes of it being submitted, then that request times out and is dropped. In these cases, you must resubmit the request later.

Scalable and Non-Scalable Queries

The getSessions API allows the client to perform custom queries on the metadata collected for each session. This API allows the client to form custom queries that can filter on any of several important data fields in the session metadata. It also provides a flexible way to combine those queries to perform powerful metadata searches.

Note If this custom query mechanism is misused, it can have serious performance impacts on not only the query itself, but the entire MediaSense system. It is normal to store large numbers of session records depending on the traffic on the system and the retention mechanism in use. However, if a query processes a large amount of data, it takes several minutes to complete. It could also take resources away from the system that are used to process new recordings and metadata, not to mention from other large queries which are executing at the same time. If, for example, two simultaneous queries both require a large amount of temporary database space in order to execute. One of them may fail due to lack of space and there are no guarantees as to which one fails.

To avoid this problem, it is important that clients form scalable queries. A scalable query is a query whose performance remains constant regardless of how much data is in the database. An example of a scalable query is: "Find all sessions whose start date or time is between 4 and 5 o'clock today." No matter how many sessions are in the database, the number of sessions between 4 and 5 o'clock today remains fairly stable and small. An example of a non-scalable query is: "Find all sessions." The number of sessions returned is proportional to the number of sessions in the database. If there are many stored sessions, then the results are huge. It takes much time to complete and impacts resources needed by other functions of the MediaSense system.

You can protect the system against queries which return an excessive number of results by using the minSessionStartDate and the maxSessionStartDate parameters in the applicable wrapper APIs. For more information on using these parameters, see the applicable session query APIs.

For more information, see Avoid Non-Scalable Queries , on page 96.

Avoid Non-Scalable Queries

If you misuse the custom query mechanism, it can have serious performance impacts on not only the query itself, but the entire MediaSense system. The entire MediaSense system is impacted because it is possible to store large numbers of session records depending on the traffic on the system and the retention mechanism in use. This system performance behavior is normal. However, if a query processes a large amount of data, it could take up to several minutes to complete. It could also take resources away from the system that are used to process new recordings and metadata.

To avoid this problem, clients must form scalable queries. A scalable query is a query whose performance remains constant regardless of how much data is in the database. An example of a scalable query is: "Find all sessions whose start date and time is between 4 and 5 o'clock today." No matter how many sessions are in the database, the number of sessions between 4 and 5 o'clock today remain fairly stable and small. An example of a non-scalable query is: "Find all sessions." The number of sessions returned is proportional to the number of sessions in the database. If there are many stored sessions, then the results are huge. It takes much time to complete and impacts resources needed by other functions of the MediaSense system.

While there is no direct way to issue a "Find all sessions" query with the getSessions API, you can form a custom query to get almost the same result. For example, because all sessions have at least one track whose media type is "Audio," you could submit a query like: "Find all sessions whose media type is Audio." While this query cannot be prevented, it can be the source of serious performance impact on the system. Although a specific value of media type is being specified ("Audio"), the value is not restrictive enough. The performance is impacted because most, if not all, sessions have a media type of "Audio." If there are many sessions saved in the database, then the query has to process all these sessions. This query is not scalable.

Note It does not help to use pagination to specify a small page size. The page is obtained only after the full result set is processed, which is too late.

A better approach is to reduce the scope of the query by reducing the number of sessions processed. In this case, perhaps what the client really wants is the last few sessions that occurred. You can reduce the number of sessions by: "Find all sessions whose start date and time is greater than one hour ago." This query returns a smaller set of results. More importantly, it returns roughly the same number of sessions regardless of how many sessions are in the database. This query is scalable.

If only the last 10 sessions are required, the client could also add the qualifier to sort the results on the start date and time field and specify pagination that displays the last 10 rows. Because the underlying result set is small, this qualifier is not a costly operation.

It is not always a problem to filter on media type. For example, the following query is scalable: "Find all sessions whose start date and time is during last Tuesday and whose media type is Video." First, narrow the overall result set by setting a time window. So the query has to search through only that small set of sessions to find the ones whose media type is "Video." In this case, the media type field does not affect scalability because it is combined with a scalable filter.

Different fields in the metadata have different degrees of scalability when used to filter sessions. Use the following table as a guide:

Table 1: Field Name and Relative Scalability

Field name	Relative scalability	Notes
sessionId	Scalable	This value is unique. If you query for a session that has a specific id, you get only one.
sessionStartDate	Scalable	Use with care. If you specify an unreasonably large window size, you still return a significant portion of the sessions in the database. Narrow the time window as much as possible.
maxSessionStartDate	Scalable	When used with minSessionStartDate this parameter allows you to specify the upper limit of a time window. Make sure that you use a reasonably sized window to limit the number of results. If this parameter is missing, then minSessionStartDate is required and the time window is assumed to be 2 hours.
minSessionStartDate	Scalable	When used with maxSessionStartDate this parameter allows you to specify the lower limit of a time window. Make sure that you use a reasonably sized window to limit the number of results. If this parameter is missing, then maxSessionStartDate is required and the time window is assumed to be 2 hours.

Field name	Relative scalability	Notes
xrefCi	Scalable	This value is unique and takes a while to cycle. When filtering using a specific xrefCi value, you get at most just a few sessions.
pruned	Mostly not scalable	This is a boolean field. If you select "false", you are likely to get the vast majority of sessions in the database. If you select "true", the result set is small and scalable.
tagType sessionState mediaType	Mostly not scalable	Only two to five possible values. For values like "Audio" for mediaType or "Completed" for sessionState, the number of sessions returned is proportional to the number of sessions that are in the database. Other values may return smaller result sets, but they will be proportional to the number of sessions in the database.
tagName	Not scalable	If tags are used, it is likely that the same tag names may be repeated across many sessions. To the degree that tags are used, the number of results returned that contain a specific tag may be proportional to the number of sessions in the database.
deviceRef	Not scalable	The extension numbers found in the deviceRef field tend to appear in many sessions. Numbers that are heavily used tend to be associated with many sessions. This number may be proportional to the number of sessions in the database.
sessionDuration	Not scalable	Although the duration can vary somewhat, a simple query like "Find all session whose duration is less than 2 minutes" can return a large portion of the sessions in the database.
nodeIPAddress partition	Not scalable	The number of partitions or nodes is relatively small. So accessing the sessions associated with any specific partition or server (node) returns several sessions that are proportional to the number of sessions in the database.

Field and parameter connectors affect scalability. See the following table:

Table 2: Connector and Relative Scalability

Connector	Relative scalability	Notes
and	Scalable	This connector narrows the scope of the filter so the result set generally gets smaller.
or	Not scalable	This connector widens the scope of the filter so the result set gets larger. Its use should be limited to connecting filters on scalable fields using scalable operators.

When you define a query filter, always be sure to use a scalable filter. If you must filter on a non-scalable field, always use the "and" connector to connect it to a scalable filter that narrows the result set.

The following are a few examples of how client applications can use the MediaSense API effectively without impacting the performance of the system.

To track the progress of a session, a client application may take advantage of both the query and event mechanism of MediaSense. Initially the client must gather the state of the sessions it is interested in. This gathering of the state of the sessions can be done by an initial query such as "Find all sessions containing extension number nnnn that are active that were started in the last day." This query narrows the number of sessions to only those sessions that started in the last day and thus will be scalable. It then quickly finds the small number of those sessions that contain the specified extension and that are active.

At the same time, the client should be subscribed to application events. This way, as the session changes state, the client can track its progress.

This is similar to the previous scenario. Make an initial query to establish the current set of active sessions: "Find all sessions that are active that started in the last day." One day should be plenty of time to ensure that you find all active sessions without taking too much system resources searching through all sessions in the database.

The client can continue to monitor the progress of the active sessions by using the event mechanism. If a new session is created, an event notifies the client so that it can begin to track it as well. Likewise when sessions close, they can be removed from the client's list of tracked sessions. The important point is that the client must do a major database query to initialize. This minimizes the impact to the system resources.

When a system is set up to use Recording Priority retention mode, the sessions are pruned based on how long they were stored. However, the metadata associated with those sessions is not removed. It is preserved so that the administrator might review before it is deleted. The expected sequence is that the client application can request a list of "pruned" sessions. It may then display the metadata of interest from those sessions. Finally the user can select one or more of the pruned sessions to be deleted.

You can do the first step with a query like: "Find all sessions whose pruned status is 'true' that were started in the last two days." Use a reasonable time window to make the query scalable. Because the pruning uses an age-based mechanism, look at least a day before today. If this is not being done on a daily basis, you may want to limit the time window to about a day. This depends upon traffic at the site. You do not want the query returning more than about 100 sessions at a time. This may require some testing.

The client should then rerun this operation for each prior day until you have searched back to the last day you previously performed this operation. The client may then present the list to the user and ask that they select the sessions to be deleted (possibly giving a "delete all" option as well). The client uses the deleteSessions method of the API to submit a job to delete the selected sessions. There is no scalability issue in this case as the deleteSessions method serializes the deletion of each session.

To monitor the progress of the deletions, the client uses the event mechanism. As each session is deleted, a session event is generated with an event type of "deleted."

CHAPTER 9

System Information

Introduction

Use these APIs to obtain the MediaSense API version information. You do not need to sign in to use this API. The getAPIVersion API returns a floating point version number for the APIs running in MediaSense deployments.

getAPIVersion

Use this API to retrieve the current version of the APIs running on the system. The version number starts at 1.0 and is not tied to the product version. The version number is x.y and not x.y.z (for example, it is 1.0 and not 1.0.1).

URI

```
https://<host>:<port>/ora/infoService/info/getAPIVersion
```

HTTP Method

GET

Parameter

apiVersion: It is an output string. The version number of the APIs currently running on the system. Although this version is returned as a string, it is also a floating point number. A relatively smaller numerical value indicates an older version. A relatively larger numerical value indicates a newer version.

Example

HTTPS GET:

```
https://10.10.10.10:8440/ora/infoService/info/
getAPIVersion
```

Response:

```
{
    "responseMessage": "Success: Your request was successfully completed.",
    "responseCode": 2000,
    "responseBody": {
        "apiVersion": "1.4"
    }
}
```

getSystemTime

Use this API to retrieve the current time of the system. The time is returned in milliseconds, since Jan 1, 1970 GMT.

As all MediaSense servers are configured as NTP clients, any differences between them are expected to be small. Therefore only a single system time is returned, and this time applies to all servers in the MediaSense cluster.

URI

`https://<host>:<port>/ora/infoService/info/getSystemTime`

HTTP Method

GET

Parameter

currentSystemTime: It is an output integer. The system time when the request was serviced; shown in the number of milliseconds since Jan 1, 1970 GMT.

Example

HTTPS GET:

```
https://10.194.118.72:8440/ora/infoService/info/
getSystemTime
```

Headers:

```
JSESSIONID: <the jsessionId received from a signIn
request>
```

Response:

```
{
    "responseMessage": "Success: Your request was
     successfully completed.",
    "responseCode": 2000,
    "responseBody": {
        "currentSystemTime": "1301145648258"
        }
}
```

getSessionsResponseSchema

Use this API to retrieve the current version of the getSessions response schema running on the system.

Note The "required" tag in the schema means that the parent property has to exist in the actual response. If "required" is not specified or it is set to false, then that property may not be returned by the response.

MediaSense uses "searchable" a little differently than the json schema draft; the node consists of an object that is an array of all possible operations. If the "searchable" value exists, then the corresponding property is searchable. If it doesn't exist or is set to false, then the property is not searchable.

The array defines properties (such as equals, greaterThan, lessThan, and between) by which the object is searchable. The between property has lowerBound and upperBound values.

URI

`https://<host>:<port>/ora/infoService/info/getSessionsResponseSchema`

HTTP Method

GET

Parameter

None

Example

HTTPS GET:

```
https://10.10.10.10:8440/ora/infoService/info/
getSessionsResponseSchema
```

Response:

```
{
 "type":"object",
 "$schema": "http://json-schema.org/draft-03/schema",
 "apiVersion": "string representation of api version",
 "properties":
   {"responseCode":
     {"type":"number",
      "required":true },
    "responseMessage":
     {"type":"string",
      "required":true }
    "responseBody":
     {"type":"object",
      "required":false
      "properties":
       {"sessions":
         {"type":"array",
          "required":true,
          "items":
           {"type":"object",
            "required":true,
            "properties":
             {"callControllerIP":
               {"type":"string",
                "required":true },
              "callControllerType":
               {"type":"string",
                "required":true },
              "ccid":
               {"type":"string",
                "required":false,
                "searchable":
                 {"operations": ["equals"] }
             },
            "sessionDuration":
             {"type":"number",
              "required":false,
              "minimum":0,
              "searchable":
               {"operations": ["equals", "lessThan",
                  "greaterThan"] }
             },
"sessionId":
             {"type":"string",
              "required":true },
            "sessionStartDate":
             {"type":"number",
              "required":true,
              "minimum":0,
              "searchable":
               {"operations": ["between"] }
             },
            "sessionState":
             {"description":"string values are ACTIVE,
               CLOSED_NORMAL, DELETED, and CLOSED_ERROR",
             "type":"string",
             "required":true,
             "searchable":
              {"operations": ["equals"] }
             },
           "tags":
            {"type":"array",
             "required":false,
             "items":
              {"type":"object",
               "required":true,
               "properties":
                {"tagCreateDate":
                  {"type":"number",
                   "required":true },
                 "tagName":
                  {"type":"string",
                   "required":true,
                   "searchable":
                    {"operations": ["equals", "contains",
                       "startsWith", "endsWith"] }
                  },
                 "tagOffset":
                  {"type":"number",
                   "required":false },
                 "tagType":
                  {"description":"string values are
                    SYSTEM_DEFINED and USER_DEFINED",
                   "type":"string",
                   "required"true,
                   "searchable":
                    {"operations": ["equals"] }
                   }
                 }
               },
 "tracks":
              {"type":"array",
               "required":true,
               "items":
                {"type":"object",
                 "required":true,
                 "properties":
                  {"codec":
                    {"type":"string",
                     "required":true },
                   "downloadUrl":
                    {"type":"string",
                     "required":false },
                   "participants":
                    {"type":"array",
                     "required":true,
                     "items":
                      {"type":"object",
                       "required":true,
                       "properties":
                         {"deviceId":
                           {"type":"string",
                            "required":true },
                          "deviceRef":
                           {"type":"string",
                            "required":true,
                            "searchable":
                             {"operations": ["equals",
                              "contains", "startsWith",
                              "endsWith", "between"] }
                           },
                          "isConference":
                            {"type":"boolean",
                             "required":true },
                          "participantDuration":
                            {"type":"number",
                             "required":false },
                          "participantStartDate":
                            {"type":"number",
                             "required":true },
                          "xRefCi":
                            {"type":"string",
                             "required":false,
                             "searchable":
                               {"operations": ["equals"] }
                            }
                          }
                        }
                      },

               "tags":
                 {"type":"array",
                  "required":false,
                  "items":
                   {"type":"object",
                    "required":true,
                    "properties":
                     {"tagCreateDate":
                       {"type":"number",
                        "required":true },
                      "tagName":
                       {"type":"string",
                        "required":true },
                      "tagOffset":
                       {"type":"number",
                        "required":false },
                      "tagType":
                       {"description":"string values are
                        SYSTEM_DEFINED and USER_DEFINED",
                        "type":"string",
                        "required":true, }
                   }
                 }
               },
"trackDuration":
```

```
                  {"type":"number",
                   "required":false },
                 "trackMediaType":
                  {"description":"string values are AUDIO and
                    VIDEO",
                   "required":true,
                   "type":"string" },
                 "trackNumber":
                  {"type":"number",
                   "required"true },
                 "trackStartDate":
                  {"type":"number",
                   "required":true }
                }
              }
            },
          "urls":
            {"type":"object",
             "required":false,
             "properties":
              {"mp4Url":
                {"type":"string",
                 "required":false },
               "rtspUrl":
                {"type":"string",
                 "required":false },
               "wavUrl":
                {"type":"string",
                 "required":false }
              }
            }
          }
        }
      }
    }
  }
 }
}
```

CHAPTER 10

User Authentication APIs

Introduction

MediaSense enables third-party developers to configure application users that allow third party applications to authenticate themselves. All access to the MediaSense API requires authentication. The first step for any application is to authenticate itself. Upon successful authentication, a JSESSION token is returned. This token must be passed in all subsequent requests until the application signs out. The JSESSIONID is also required in the signOut API request.

If third-party applications are designed to issue API requests to both the primary and the secondary servers, then they must sign in to each node independently, and always use the corresponding JSESSIONID with the server that returned the request.

The JESSIONID expires when the user logs out or after 30 minutes of inactivity, whichever comes first.

If the requests are made from a browser, then the browser automatically handles the session for the clients.

Note When your client logs in to the third-party client to use the MediaSense platform, the third-party software must perform authentication as dictated by that software application.

signIn

Use this API for user authentication/sign-in. The API checks the authentication of the given username and password. On successful sign-in, a JSESSIONID is returned as a cookie in the header of the HTTP response. This JSESSIONID serves as a token of authentication and must be passed in all subsequent requests. If the sign-in is successful, the API checks the configuration database for the parameters set by the administrator that are included in the JSON response of the API. The parameters that have not been set by the administrator are excluded from the JSON response.

URI

```
https://<host>:<port>/ora/authenticationService/authentication/signIn
```

HTTP Method

POST

Parameters

- archiveSearchEnabled— It is a boolean. If the value is true, the archive search is enabled in MediaSense Search and Play.
- authenticationProviders— It is an optional input array of strings. The identifiers of the services to use for authenticating the user. Enumeration values are:
    - AXL
    - Finesse

  If this parameter is unspecified, the system attempts to authenticate with the providers in the order listed above and returns when the first one succeeds.
- firstNameDisplayConfig— It is a boolean. If the value is true, the first name appears as an advanced search field and is displayed as part of agent information in MediaSense Search and Play.
- inbrowserPlaybackEnabled— It is a boolean. It indicates that In-browser playback is enabled in MediaSense Search and Play.
- lastNameDisplayConfig— It is a boolean. If the value is true, the last name appears as an advanced search field and is displayed as part of agent information in MediaSense Search and Play.
- lineNameDisplayConfig— It is a boolean. If the value is true, the Unified Communications Manager line name appears as an advanced search field and is displayed as part of agent information in MediaSense Search and Play.
- loginIdDisplayConfig— It is a boolean. If the value is true, the login identifier appears as advanced search field and is displayed as part of agent information in MediaSense Search and Play.
- loginNameDisplayConfig— It is a boolean. If the value is true, the login name appears as advanced search field and is displayed as part of agent information in MediaSense Search and Play.
- password— It is a required input string. The password of the Unified Communications Manager user who is provisioned as a MediaSense user. All Unified Communications Manager password restrictions apply. Because Unified Border Element uses SSH authentication, MediaSense uses Unified Communications Manager authentication for all MediaSense users. It is case sensitive.
- username— It is a required input string. The username of the Unified Communications Manager user who is provisioned as a MediaSense user. Unified Communications Manager uses "User ID" while MediaSense uses username (non case-sensitive). All Unified Communications Manager User ID restrictions apply. Because Unified Border Element uses SSH authentication, MediaSense uses Unified Communication Manager authentication for all MediaSense users.

Note

When an API user's password changes in Unified Communications Manager, it may take several minutes for that change to be reflected in MediaSense. A successful response also returns a JSESSIONID inside a cookie.

Examples

Example 1

To signIn a user with username " myUser " and password " cisco " on the server 10.194.118.1 at port 8440:

Since no authentication provider has been specified, this request will try to authenticate against all possible configured authentication providers including AXL and Finesse.

HTTPS POST:

```
https://10.194.118.1:8440/ora/authenticationService/authentication/signIn
```

Headers:

```
Content-Type: application/json
```

Body:

```
{
   "requestParameters": {
                           "username":"myUser",
                           "password":"cisco"
                      }
}
```

Example 2

To signIn a Finesse supervisor with username " myUser " and password " cisco " on the server 10.194.118.1 at port 8440:

HTTPS POST:

```
https://10.194.118.1:8440/ora/authenticationService/authentication/signIn
```

Header:

```
Content-Type: application/json
```

Body

```
{
   "requestParameters": {
                           "username":"myUser",
                           "password":"cisco"
                        },
                      "authenticationProviders": ["FINESSE"]

}
```

Example 3: Response Parameters

```
{
    "responseMessage": "Success: Your request was successfully completed.",
    "responseCode": 2000,
    "inbrowserPlaybackEnabled": "true",
    "lineNameDisplayConfig": "true",
    "archiveSearchEnabled": "true",
    "agentDataDisplayConfig": {
        "loginNameDisplayConfig": "true",
        "loginIdDisplayConfig": "true",
        "lastNameDisplayConfig": "true",
        "firstNameDisplayConfig": "true"
    }
}
```

signOut

This API is used to sign out the user from MediaSense. The user requires no parameter other than JSESSIONID to sign out.

URI

`https://<host>:<port>/ora/authenticationService/authentication/signOut`

HTTP Method

POST

Request Parameters

None

Examples

To signOut a user who was previously signed in:

HTTPS POST:

```
https://10.194.118.1:8440/ora/authenticationService/
authentication/signOut
```

Headers:

```
JSESSIONID: <the jsessionId received from a previous
signIn request>

Content-Type: application/json
```

The user identified by the JSESSIONID is signed out.

CHAPTER 11

Shared Parameters

Introduction

MediaSense APIs use the term parameter to describe items with a name and value in JSON objects. We acknowledge that these items are referred to by various terms in different programming interfaces.

The parameters are classified as follows:

- *Required Input* parameters are user-defined values, are necessary, and are supplied by the user to the system when the API or event is issued.
- *Optional Input* parameters are user-defined values and may be supplied by the user to the system when the API or event is issued. These parameter values are supplied only when users want to supply them.
- *Output* parameters have values that are system-generated and are the results of issuing the API or event. These values are calculated according to the input parameter values and the APIs or events that use them.

Common Parameters for All APIs

The following parameters are common across all APIs.

detail

- Output string.
- Provides information about response codes.

JSESSIONID

- Both an output string and a required input string.

- Header parameter.
- Session identifier and serves as a means of authentication for subsequent requests.
- Expires when the user explicitly logs out or within 30 minutes of inactivity, whichever comes first.

responseCode

- Output integer.
- Corresponding error code.

responseMessage

- Output string.
- Description of the responseCode for troubleshooting purposes.

Shared Parameters

byFieldName

- Optional input string.
- Required input string with sortParameters.
- Enumeration values differ based on the API that is using this parameter.

callControllerIP

- Output string.
- When the callControllerType = Cisco-SIPGateway, it is the IP address of the gateway which originated the recording.
- When the callControllerType = Cisco-Unified Communications Manager, it is the IP address of the Unified Communications Manager device which originated the recording.
- When the callControllerType = direct, it is the IP address of the IOS or Unified Ccommunications Manager device which is communicating with MediaSense.

callControllerType

- Output string.
- Enumeration values (case-sensitive, identified in bold)
    - **Cisco-SIP Gateway** (calls from Unified Border Element).
    - **Cisco-Unified Communications Manager** (calls forked from Unified Communications Manager).
    - **direct** (direct outbound/inbound calls initiated by MediaSense).

- ◦ **Cisco-Unified Communications Manager-Gateway** (calls forked from a gateway under the control of Unified Communications Manager)

ccid

- Output string.
- Optional input string when getSessions is used.
- Required input string when getSessionsByCCID is used.
- Call Correlation ID.
- Always present for calls forked from the Unified Border Element Gateway, but may not be present for calls forked from Unified Communications Manager phones.
- Used to identify recording sessions which are part of the same call.
- Corresponds to the "Cisco-guid" field which is provided by the Unified Border Element Gateway, and can therefore also be used to correlate recording sessions with call information from other components in the solution.

codec

- Output string.
- Codec of the track.
- Contains the standard name from the codec's SDP definition.

Note The g.711 µ-law and a-law codecs are known in the SDP as "PCMU" and "PCMA" respectively. The AAC-LD codec is known as MP4A-LATM.

deviceId

- Output string.
- Optional input string when getSessions is used.
- Unique identifier of the device.

deviceRef

- Output string.
- Phone number of each device.
- When either party on a call is a conference bridge, deviceRef shows "Anonymous" for Unified Border Element forking or "b" followed by a string of digits for BiB forking.

downloadUrl

- Output string.

- URL that is used to download the recording in the raw format.
- Conditionally present only when all three of the following conditions are met:
    - If the API sessionState = CLOSED_NORMAL or the eventAction = ENDED
    - and if track mediaType = AUDIO
    - and if trackSize > 0 The downloadUrl parameter is not available for other sessions in ACTIVE, DELETED, or CLOSED_ERROR states. The downloadUrl parameter should be treated by clients as an opaque string. Client code should not depend on its format or structure in any way other than to assume that it contains a fully-formed HTTPS URL. Clients can add the URL parameter "?timeout=n" to indicate that MediaSense should try to write to the socket for at least n seconds before it aborts the download. The default value is n = 5 seconds.

errorCode

- Output integer.
- Appears when there is an error in a report for a particular session.

errorDetail

- Output string.
- When the session state is CLOSED_ERROR, the *errorDetail* field will have one of the following values:
    - MEDIA_SERVER_ERROR: The Call Control server receives an error response from the Media (recording) server for an open-session or close-session request.
    - MEDIA_SERVER_TIMEOUT: The Call Control server is timed out waiting for response from the Media (recording) server for an open-session or close-session request.
    - SIP_SIGNALING_ERROR: The Call Control server detects a SIP signaling error. For example, a missing ACK.
    - SIP_CANCEL_RECEIVED: The recording was canceled by the Call Control server, such as CANCEL or premature BYE.
    - NO_MEDIA_RECEIVED: The session was closed successfully, however, all tracks have size as zero.
    - ORPHANED: The session was orphaned. This can occur if (for example) the session was forcibly closed after service restart.
    - UNSUPPORTED_CODEC: The codec is not supported.

eventAction

- Output string.
- The eventAction (sessionEvent) parameter indicates if a new recording session started or if an existing recording session was updated, ended, deleted or pruned. Enumeration values:
    - STARTED
    - UPDATED

  - ENDED
  - DELETED
  - PRUNED

- The eventAction (storageThresholdEvent) parameter is sent each time the storage disk space reaches the specified threshold.
- The eventAction (tagEvent) parameter indicates if a new tag is added or existing tag is updated or deleted. Enumeration values:
  - ADDED
  - DELETED

eventType

- Output string.
- Events triggered for the corresponding MediaSense API.
- Enumeration values:
  - SESSION_EVENT
  - STORAGE_THRESHOLD_EVENT
  - TAG_EVENT

fieldConditions

- Required input array of strings.
- Conditions specified for the queried field.
- Specify multiple conditions for a field name.
- Contains details on the fieldOperator, fieldValues, and fieldConnector.
- Enumeration values differ based on the APIs using this parameter

fieldConnector

- Optional input string when you have one field in an array.
- Required input string when you have two or more fields in an array.
- Used to logically connect a condition with the successive condition.
- Enumeration values:
  - AND
  - OR

fieldName

- Required input string.
- Name of the queried field.
- Enumeration values differ based on the APIs using this parameter.

fieldOperator

- Required input string.
- Logical operator conditions as specified in the **Values Expected for Each fieldOperator** section.
- Permitted number of operands for each operator:
    - equals = 1
    - contains = 1
    - startsWith = 1
    - endsWith = 1
    - lessThan = 1
    - greaterThan = 1
    - between = 2 (include boundary values).
    - Example: Between 11 - 14 includes 11, 12, 13, and 14. You can prepend a special negation operator (!) * to all operators other than AND and OR to negate their meaning. Examples:
        - "equals" is negated by "!equals".
        - "between" is negated by "!between" to mean all values lying outside the interval A - B.

fieldValues

- Required input array of strings.
- Number of values passed in this array depends on the fieldOperator as specified in the **Values Expected for Each fieldOperator** section.
- Does not accept regular expression (*,+ and so on)
- Enumeration values differ based on the APIs using this parameter.

firstName

- Output string.
- First name of the agent logged in to the Finesse Agent desktop.

forwardedEvent

- Omitted if events are locally-generated on the MediaSense server and if forwarding is disabled.

- Disabled by default in MediaSense deployments.
- Enumeration values:
    - TRUE = Enabled
    - FALSE (default) = Disabled

httpUrl

- Deprecated in release 10 in favor of mp4Url.
- Output string.
- HTTPS link for the session.
- Url is only present if the session state is CLOSED_NORMAL.
- Has a maximum of 512 characters.

Note Converted mp4 and wav files are deleted by the system a maximum of 2 hours after they have been created or last modified.

isConference

- output Boolean.
- Used within a Participant object.
- Indicates whether the participant is a conference bridge or an individual device.

Note In certain scenarios, one participant may represent a conference bridge.

jobDuration

- Output integer when getJobById or getJobResult is used.
- Optional input integer when getJobs is used.
- Time, in milliseconds, between the jobStartTime and the time when this job ends.

jobId

- Optional input string when getJobs is used.
- Output string when all other APIs or events in the list are used.
- Job created and managed by the job management framework.
- System-generated.
- Used along with the Job management and Job query APIs.

jobState

- Output string when getJobById or getJobResult is used.
- Optional input string when getJobs is used.
- After retrieving the job status, the applicable value is pulled from the database based on the value at the time of polling.

jobs

- Output array of job objects.
- List of jobs.
- Details for each job.

jobStartTime

- Output integer when getJobById or getJobResult is used
- Optional input integer when getJobs is used.
- Time, in milliseconds, since Jan 1, 1970 GMT when the job started.
- Time when the job starts to execute.

jobType

- Output string when getJobById or getJobResult is used.
- Optional input string when getJobs is used.
- Purpose of the job.
- Case sensitive
- Enumeration value is BULK_DELETE_SESSIONS.

lastName

- Output string.
- Last name of the agent logged in to the Finesse Agent desktop.

limit

- Required input integer.
- Number of records to be returned, starting at the specified offset.
- Required within pageParameters.
- Must be used in conjunction with offset parameter.
- By default, it returns the first 100 records.
- Any positive value greater than zero (0).

- Maximum possible value for this parameter is 1000.

lineDisplayName

- Output string.
- Name that appears on a phone as a calling party, which is entered in the **Display (Caller ID)** field in the **Directory Number Configuration** window in **Cisco Unified Call Manager Administration**.

loginId

- Output string.
- Login Id of the agent logged in to the Finesse Agent desktop.

loginIdDomain

- Output string.
- Login Id domain of the agent logged in to the Finesse Agent desktop.

loginName

- Output string.
- Login name of the agent logged in to the Finesse Agent desktop.

maxSessionStartDate

- Optional input integer.
- Required input integer when no value is specific for minSessionStartDate.
- Number of milliseconds since Jan 1, 1970 GMT when the session recording started, or more precisely, when the first track for this session began recording.
- If either minSessionStartDate or maxSessionStartDate is missing, it is assumed to be 2 hours in that direction.

mediaType

- Output string and a required input string.
- Describes the type of media being established.
- Case sensitive.
- Enumeration values are:
    - AUDIO
    - VIDEO

mediaStreams

- Required input array of media types.
- media streams being established.
- Each stream is a media type.
- mediaStreams is an array of media types.
- A recording must have at least one media stream
- Array is not ordered.

minSessionStartDate

- Optional input integer.
- Number of milliseconds since Jan 1, 1970 GMT when the session recording started, or more precisely, when the first track for this session began recording.
- If either minSessionStartDate or maxSessionStartDate is missing, it is assumed to be 2 hours in that direction.

mp4Url

- Output string.
- mp4 link for the session.
- Has a maximum of 512 characters.
- mp4Url is only present when the session state is CLOSED_NORMAL and may return the following HTTPS response codes:
    - 200 OK
    - 404 Not Found
    - 500 Internal Server Error
    - 503 Service Unavailable (When Cisco MediaSense Media Service is unavailable) Sessions are converted to mp4 the first time the URL is accessed, so there may be a delay before the results are returned the first time.

Note Mp4 files are generated when the URL is accessed and then cached for a period of time. If the file needs to be generated, there may be a delay before this URL request responds.

Converted mp4 files are deleted by the system a maximum of 2 hours after they have been created or last modified.

nodeId

- Output integer.

- System-defined numeric value.

nodeIPAddress

- Output string.
- IP address of the Cisco MediaSense server.
- Has a maximum of 54 characters.

offset

- Optional input integer.
- Becomes a required input integer within pageParameters.
- First record to be returned.
- Must be used in conjunction with limit parameter.
- Integer value starts at zero (0)

order

- Optional input string.
- Becomes a required input string within sortParameters.
- Case sensitive.
- Enumeration values are:
    - ASC= Ascending: Sorts results with the lowest value first.
    - DESC=Descending: Sorts results with the highest value first.

pageParameters

- Optional input.
- JSON object.
- Allows the client to request a specific subset of a request result representation.
- Specifies the elements to return.
- If used, it returns the first 1000 sessions.
- Must also specify the limit and offset parameters.

paramConnector

- Optional input string when you have only one field in the array.
- Required input string when you have two or more fields in the array.
- Case sensitive.

- Specify the logical connection between two elements in the RequestParameters array.
- Enumeration values are:
    - AND
    - OR

partition

- Output string.
- Name of the datastore using storage space.
- Has a maximum of 128 characters.

participantDuration

- Output integer.
- Number of milliseconds that the participant was active in the session.

Note The durations are derived from SIP call control information, and are typically a few seconds longer than the actual media duration.

participantInformation

- Output JSON array of participant information.
- Information contains details on loginId, lastName, firstName, loginIdDomain, and loginName parameters.

participants

- Output JSON array of participant objects.
- Each object in turn contains details on the deviceRef, participantStartDate, participantDuration, isConference, and the xRefCi parameters.

participantStartDate

- Output integer.
- Number of milliseconds since Jan 1, 1970 GMT when this track's recording started.

rtspUrl

- Output string.
- Reference to the entire session, which can contain multiple tracks.
- To access a particular track, suffix the rtspUrl with "-TRACK0" or "-TRACK1". Both tracks in a given session open and close simultaneously.

- URLs stay constant for the life of the session.
- Has a maximum of 512 characters.
- Enumeration values are:
    - If eventAction = STARTED/UPDATED, then rtspUrl = monitor (used to monitor the currently-recorded stream)
    - If eventAction = ENDED, then rtspUrl = play (used to play the stored media.)

sessions

- Output JSON array of session objects.
- Each object contains values for sessionId, tracks, URLs, sessionStartDate, and sessionDuration.

sessionDuration

- Output integer.
- Number of milliseconds that the session lasted.

Note These durations are derived from SIP call control information, and are typically a few seconds longer than the actual media duration.

sessionId

- System-generated identifier for a session.
- Unique across all MediaSense servers and has a maximum of 128 characters. A session is a unit of the capture service, where one session is converted i
- Session is a unit of the capture service, where one session is converted into one MP4 file.
- Required input string for startRecording, stopRecording, addSessionTag, convertSession, and getSessionBySessionId.
- Output string for the remaining APIs and events.

sessionIdList

- Accepts a list of multiple comma-separated sessionIds in the request body, but currently, only the first sessionId will be considered to return the strongly associated sessionIds.
- Additional sessionIds will be ignored.
- Made of the *List* type for future enhancement.

sessionStartDate

- Output integer for all APIs and events.

- Required input integer for getSessions.
- Number of milliseconds since Jan 1, 1970 GMT when the session recording started, or more precisely, when the first track for this session began recording.

sortParameters

- Optional input JSON array.
- Specifies how the returned result should be sorted.
- Default sorting is by the startDate with the newest session first.

sessionState

- Output string.
- State of the session.
- Enumeration values are:
    - ACTIVE = At least one session within the session is active.
    - CLOSED_NORMAL = All sessions within the session are closed without errors.
    - DELETED = User-deleted session or system-pruned session
    - CLOSED_ERROR = At least one session within the session has errors (could not be recorded).

subscriptionFilters

- Output JSON array and an optional input JSON array.
- Specifies a list of events or a list of event categories.
- Depending on which API it is used in, its meaning varies.

1. The events to which the client wants to subscribe.
2. Within the unsubscribeFromEvents API:
    1. When used in the request: The events that the client wants to **unsubscribe from**.
    2. When used in the response: The events that the **client is now subscribed to**, after performing the unsubscription.
3. The events, to which the client is currently **subscribed**. Case-sensitive **Enumeration Values** include: * ALL_EVENTS (category)
    - RECORDING_EVENTS (category)
        - SESSION_STARTED_EVENT
        - SESSION_UPDATED_EVENT
        - SESSION_ENDED_EVENT
    - CLEANUP_EVENTS (category)
        - SESSION_DELETED_EVENT

    - SESSION_PRUNED_EVENT

- TAG_EVENTS (category)
    - TAG_ADDED_EVENT
    - TAG_DELETED_EVENT
    - TAG_UPDATED_EVENT
- STORAGE_EVENTS (category)
    - ENTER_LOW_STORAGE_ SPACE_EVENT
    - EXIT_LOW_STORAGE_ SPACE_EVENT
    - ENTER_CRITICAL_STORAGE_ SPACE_EVENT
    - EXIT_CRITICAL_STORAGE_ SPACE_EVENT
    - ENTER_EMERGENCY_STORAGE_ SPACE_EVENT
    - EXIT_EMERGENCY_STORAGE_ SPACE_EVENT

The client can use the category name or the specific event names. If a category is used, the client is subscribed to all events in that category.

subscriptionId

- Output string for subscribeToEvents.
- Required input string for all other APIs and events in the list.
- System-generated ID that is received from the subscribeRecordingEvent API.

subscriptionStatus

- Output string.
- Response that is received from the verifyEventSubscription API or the verifyRecordingSubscription API.
- Enumeration values are:
    - ACTIVE
    - INACTIVE

subscriptionType

- Output string.
- Type of subscription.
- For server-based clients, the type is HTTP.

subscriptionUri

- Required input string.
- URI where event notifications are sent to server-based clients.
- Subscription is keyed by the subscriptionUri parameter.
- Only one subscription is allowed for any given subscriptionUri.

tags

- Output JSON array of tag objects.
- Each object includes information on the tagName, tagType, tagCreateDate, and tagOffset parameters.
- Has a maximum of 255 characters.

tagCreateDate

- Output integer.
- Number of milliseconds since Jan 1, 1970 GMT when this tag was created.

tagName

- Output string.
- Name that is used to label a recording, which is not case sensitive.
- Has a maximum limit of 255 characters.
- If tagtype = SYSTEM_DEFINED, then the enumeration values for tagName are as follows:
    - If the tag is at the session level:
        - Pause
        - Resume
    - If the tag is at the track level:
        - TrackActive
        - TrackInactive

tagNameRegEx

- Optional input string.
- Java regular expression which only applies to tag events.
- Additive string, that is, MediaSense only sends a tagEvent to the subscriber when the event matches the actions that are requested by the subscriber and when the tagName matches the subscriber's optional regular expression specified by this parameter.

See http://download.oracle.com/javase/tutorial/essential/regex/ for more information on Java regular expressions.

tagOffset

- Optional integer for addSessionTag.
- Number of milliseconds from the start of session for this tag.
- Use tagOffset to specify a point in the recording. For example, at the 10th second from the beginning of a specific recording, an unexpected event occurred.) It is an output integer for all other APIs and events in the list.
- Offset is calculated from the start of the session or track (regardless of the tag being present at the track level or the session level).
- If no offset is specified, the tag is applied to the entire session and not to any specific point in the session.

tagType

- Output string.
- Enumeration values are:
  - SYSTEM_DEFINED = Default. Used for user-triggered system events (Pause and Resume, TrackInactive and TrackActive).
  - USER_DEFINED = Users can create a tag (addSessionTag, on page 57 API). However, all tag types are system-defined and users cannot redefine the value of any tagType.

trackDuration

- Output integer.
- Number of milliseconds that the track lasted.

Note These durations are derived from SIP call control information, and are typically a few seconds longer than the actual media duration.

trackMediaType

- Output integer.
- Enumeration values are:
  - AUDIO
  - VIDEO

trackNumber

- Output integer.
- System-generated unique identifier of the track.

For Unified Communications Manager, track 0 contains media streamed from the forking device and track 1 contains media streamed to the forking device.

In a Unified Comminications Manager sessionEvent, each track is associated with one or more participants generating the media for each track.

For Unified Border Element, The trackNumber assignment is arbitrary and is not based on forking devices. The order in which elements appear in the track array is arbitrary and has no particular meaning.

tracks

- Output JSON array of track objects.
- Each object in turn includes information on the trackNumber, trackStartDate, trackDuration, trackMediaType, and participants parameters.

trackStartDate

- Output integer.
- Number of milliseconds since Jan 1, 1970 GMT when the track recording started (when the Call Control Service received the 200 OK message from Unified Communications Manager or Unified Border Element).
- MediaSense sends an acknowledgment (ACK) back to Unified Communications Manager or Unified Border Element accordingly.

urls

- Output JSON object.
- Includes information on the httpUrl and the rtspUrl parameters.

wavUrl

- Output string.
- Wav link for the session.
- Has a maximum of 512 characters.
- For sessions that also include video tracks, the wavUrl will only contain the audio tracks.
- The wavUrl is only present when the session state is CLOSED_NORMAL and may return the following HTTPS Response Codes:
    - 200 OK
    - 404 Not Found
    - 500 Internal Server Error
    - 503 Service Unavailable (When MediaSense Media Service is unavailable)

Note Wav files are generated when the URL is accessed and then cached for a period of time. If the file needs to be generated, there may be a delay before this URL request responds.

Converted wav files are deleted by the system a maximum of 2 hours after they have been created or last modified.

xRefCi

- Output string except for getSessionsByCCID, it is both an output string and an optional input string.
- The Unified Communications Manager identifier for a particular media stream.
- Do not always correspond one-to-one with the recorded tracks. However, you must use this identifier to correlate the Unified Communications Manager JTAPI events to recorded sessions.
- xRefCi is present when the callControllerType for a session = "Cisco-SIPGateway" or "Cisco-Unified Communications Manager". xRefCi is omitted when callControllerType = "direct".
- Unified Border Element does not use this identifier.

CHAPTER 12

Events Triggered

sessionEvent

This event is sent each time a new recording session is started or an existing recording session is updated or ended. It is also sent when a recording session is deleted or pruned. For deleted and pruned sessions, the sessions are batched and sent in a single event.

Method

POST

Body

JSON

Related APIs

- startRecording, on page 49
- stopRecording, on page 52

Example

```
{
    "eventType": "SESSION_EVENT",
    "eventAction": "DELETED",
    "forwardedEvent": "TRUE",
    "eventBody": {
        "sessionIds": ["Session-6-10.194.118.57-1284096640651",
        "Session-6-10.194.118.57-1284096640652"]
    }
}
{
    "eventAction": "STARTED",
    "eventBody": {
        "sessionId": "Session-6-10.194.118.57-1284096640651",
        "callControllerType": "Cisco-CUCM",
        "callControllerIP": "10.194.118.57"
        "ccid": "2728850048-0000065536-0000018373",
        "sessionStartDate": 1284096641174,
        "sessionState": "ACTIVE",
       "tracks": [
                    {
                        "trackStartDate": 1429519331647,
                        "trackDuration": 14252,
                        "codec": "PCMU",
                        "downloadUrl":
"https://10.126.135.72:8446/mma/ExportRaw?recording=7814cd5fdf06f1-TRACK1",
                        "trackNumber": 1,
                        "trackMediaType": "AUDIO",
                        "participants": [
                            {
                                "participantStartDate": 1429519331647,
                                "participantInformation":{
                                     "loginId": "ambdev",
                                     "lastName": "dev",
                                     "firstname": "ambrose",
                                     "loginIdDomain": "1",
                                     "loginName": "ambdev",
                                  },
                                "deviceRef": "11052",
                                "isConference": false,
                                "xRefCi": "29236211",
                                "participantDuration": 14252,
                                "deviceId": "trunk_to_cube150"
                            }
                        ]
                    },
                    {
                        "trackStartDate": 1429519331647,
                        "trackDuration": 14252,
                        "codec": "PCMU",
                        "downloadUrl":
"https://10.126.135.72:8446/mma/ExportRaw?recording=7814cd5fdf06f1-TRACK0",
                        "trackNumber": 0,
                        "trackMediaType": "AUDIO",
                        "participants": [
                            {
                                "participantStartDate": 1429519331647,
                                "participantInformation":{
                                     "loginId": "avmaitra",
                                     "lastName": "maitra",
                                     "firstname": "avirup",
                                     "loginIdDomain": "1",
                                     "loginName": "avmaitra",
                                  },
                                "deviceRef": "11024",
                                "lineDisplayName": "Avirup12",
                                "isConference": false,
                                "xRefCi": "29236210",
                                "participantDuration": 14252,
                                "deviceId": "SEPF0292958FA6D"
                            }
                        ]
                    }
                ],
        "urls": [
            {
                "rtspUrl": "rtsp://10.194.118.57/live/
                 Session-6-10.194.118.57-1284096640651"
            }
        ]
    },
    "eventType": "SESSION_EVENT",
    "forwardedEvent": "TRUE"
}
{
    "eventAction": "ENDED",
    "eventBody": {
        "sessionId": "Session-6-10.194.118.57-1284096640651",
        "callControllerType": "Cisco-CUCM",
        "callControllerIP": "10.194.118.57"
        "ccid": "2728850048-0000065536-0000018373",
        "sessionState": "CLOSED_NORMAL",
        "sessionStartDate": 1284096641174,
        "sessionDuration": 8394,
        "tracks": [
                    {
                        "trackStartDate": 1429519331647,
                        "trackDuration": 14252,
                        "codec": "PCMU",
                        "downloadUrl":
"https://10.126.135.72:8446/mma/ExportRaw?recording=7814cd5fdf06f1-TRACK1",
                        "trackNumber": 1,
                        "trackMediaType": "AUDIO",
                        "participants": [
                            {
                                "participantStartDate": 1429519331647,
                                "participantInformation":{
                                     "loginId": "ambdev",
                                     "lastName": "dev",
                                     "firstname": "ambrose",
                                     "loginIdDomain": "1",
                                     "loginName": "ambdev",
                                  },
                                "deviceRef": "11052",
                                "isConference": false,
                                "xRefCi": "29236211",
                                "participantDuration": 14252,
                                "deviceId": "trunk_to_cube150"
                            }
                        ]
                    },
                    {
                        "trackStartDate": 1429519331647,
                        "trackDuration": 14252,
                        "codec": "PCMU",
                        "downloadUrl":
"https://10.126.135.72:8446/mma/ExportRaw?recording=7814cd5fdf06f1-TRACK0",
                        "trackNumber": 0,
                        "trackMediaType": "AUDIO",
                        "participants": [
                            {
                                "participantStartDate": 1429519331647,
                                "participantInformation":{
                                     "loginId": "avmaitra",
                                     "lastName": "maitra",
                                     "firstname": "avirup",
                                     "loginIdDomain": "1",
                                     "loginName": "avmaitra",
                                  },
                                "deviceRef": "11024",
                                "lineDisplayName": "Avirup12",
                                "isConference": false,
                                "xRefCi": "29236210",
                                "participantDuration": 14252,
                                "deviceId": "SEPF0292958FA6D"
                            }
                        ]
                    }
                ],
        "urls": [
            {
                "rtspUrl": "rtsp://10.194.118.57/archive/
                 Session-6-10.194.118.57-1284096640651"
            }
        ]
    },
    "eventType": "SESSION_EVENT",
    "forwardedEvent": "TRUE"
}
```

storageThresholdEvent

This event is sent each time the storage disk space (that stores the recorded media) reaches various thresholds. More information about threshold values is provided in a table that follows.

Method

POST

Body

JSON

Related API

subscribeRecordingEvent(deprecated), on page 33

Example

```
{
 "eventType":"STORAGE_THRESHOLD_EVENT",
 "eventAction":"ENTER_LOW_STORAGE_SPACE",
 "forwardedEvent":"TRUE",
 "eventBody": {
                 "nodeId": 1,
                 "nodeIPAddress":"10.X.X.X",
                 "partition" : "/common",
                 "percentageUtilization": 70
               }


 }
```

Event Action Values

The threshold values percentage and implications are provided in the following table:

Threshold	Storage Percentage	Description
ENTER_LOW_ STORAGE_SPACE	Recorded media crossed the 75% storage utilization mark.	First warning to indicate that the disk storage is running into low space condition.
EXIT_LOW_ STORAGE_SPACE	Recorded media usage dropped below 70% utilization mark.	The disk storage is exiting the low storage space condition.

Threshold	Storage Percentage	Description
ENTER_CRITICAL_ STORAGE_SPACE	Recorded media crossed the 90% local storage utilization mark.	Second warning. When entering this condition, you must take action to guarantee future recording resources on this node. If operating in the Retention priority mode, new recording sessions are not accepted when you reach this threshold. In the New Recording Priority Mode, older recordings are automatically pruned to make room for new recordings.
EXIT_CRITICAL_ STORAGE_SPACE	Recorded media usage dropped below the 85% utilization mark.	The disk storage is exiting the critical storage space condition. At this point the local node is still considered to be low on resources.
ENTER_EMERGENCY_ STORAGE_SPACE	Recorded media crossed the 99% storage utilization mark.	Last warning. When entering this condition, you must take action to guarantee future recording resources on this node. If operating in the Retention priority mode, existing recording sessions are terminated when you reach this threshold.
EXIT_EMERGENCY_ STORAGE_SPACE	Recorded media usage dropped below the 97% utilization mark.	The disk storage is exiting the emergency storage space condition. At this point, the local node is still considered to be low on resources and new recording sessions are still not accepted.

tagEvent

This event is sent when a tag is added or deleted from a session. In a multi-server MediaSense deployment, the client can subscribe to only one MediaSense API service and receive notifications generated by other MediaSense services.

Method

POST

Body

JSON

Related APIs

- addSessionTag, on page 57
- pauseRecording, on page 48
- resumeRecording, on page 48
- deleteSessionTag, on page 61

Examples

The following example shows a user-defined tag that is time-bound (the tagOffset field is present). The date the tag was created is Sept 9, 2010 23:14:13 GMT. You can use an Epoch converter to see the date and time in a sensible format. The offset represents the number of milliseconds from the start of the session that the tag represents. In this case it represents five minutes.

Note MediaSense does not allow the use of tagOffset 0 (zero value) for a USER_DEFINED time-bound tag. When using a time-bound tag to start a recording, use a tagOffset of at least 1 millisecond.

```
{
    "eventType": "TAG_EVENT",
    "eventAction": "UPDATED",
    "eventBody":  {
                       "sessionId": "1234abcd5678efgh90xyz",
                       "tagName": "Customer response",
                       "tagType": "USER_DEFINED",
                       "tagCreateDate": 1284074053,
                       "tagOffset": 300000

                  }
}
```

The following example shows a system-defined tag that was added to a track within a session. The track is identified by the trackNumber.

```
{
    "eventType": "TAG_EVENT",
    "eventAction": "ADDED",
    "eventBody": {
        "sessionId": "57130d87d73f41",
        "tagCreateDate": 1309302169848,
        "tagName": "TrackInactive",
        "tagOffset": 11203,
        "tagType": "SYSTEM_DEFINED",
        "trackNumber": 1
    }
}
```