



# Cisco TelePresence Server API

Product Programming Reference Guide

**Last Updated: April 2016**

4.3



# Contents

<b>Contents</b> .....	<b>3</b>
<b>Overview</b> .....	<b>7</b>
<b>Flexible Operation Mode</b> .....	<b>8</b>
Introduction .....	8
Remotely Managed Mode API Change Summary .....	8
Design Considerations .....	12
XML-RPC Implementation .....	13
Transport .....	13
Encoding .....	13
Message Flow .....	13
Data Types and Sizes .....	15
HTTP keep-alives .....	15
API Overview .....	16
Authentication .....	16
Identifiers and Client References .....	16
Conference URI Identifiers .....	17
Participants .....	18
"Unlimited" Integers .....	20
Media Credits .....	20
Media Reservation .....	21
Enumeration .....	21
DTMF .....	23
Feedback Receivers .....	23
Data structures and enumerated types .....	26
Enumerated Types .....	26
Structs .....	32
Flexible API Command Reference .....	41
callHome.configure .....	43
callHome.query .....	44
cdrlog.enumerate .....	44
cdrlog.query .....	45
device.feature.add .....	46
device.feature.remove .....	46
device.health.query .....	46

device.network.modify .....	47
device.network.query .....	48
device.query .....	50
device.restart .....	51
device.restartlog.query .....	52
device.time.modify .....	52
device.time.query .....	53
feedbackReceiver.configure .....	53
feedbackReceiver.query .....	54
feedbackReceiver.reconfigure .....	55
feedbackReceiver.remove .....	55
feedbackReceiver.status .....	55
flex.call.status .....	56
flex.conference.create .....	57
flex.conference.deletions.enumerate .....	64
flex.conference.destroy .....	64
flex.conference.enumerate .....	64
flex.conference.getMetadata .....	67
flex.conference.modify .....	67
flex.conference.query .....	72
flex.conference.sendUserMessage .....	78
flex.conference.sendWarning .....	78
flex.conference.status .....	78
flex.licenseMode.modify .....	79
flex.licenseMode.verify .....	80
flex.participant.advanced.enumerate .....	80
flex.participant.call.disconnect .....	83
flex.participant.clearImportant .....	83
flex.participant.create .....	84
flex.participant.deletions.enumerate .....	86
flex.participant.destroy .....	87
flex.participant.enumerate .....	87
flex.participant.media.enumerate .....	89
flex.participant.modify .....	91
flex.participant.query .....	91
flex.participant.requestDiagnostics .....	93

flex.participant.requestPreview .....	99
flex.participant.sendDTMF .....	101
flex.participant.sendUserMessage .....	102
flex.participant.setImportant .....	102
flex.participant.setMute .....	102
flex.participant.status .....	103
flex.reservation.create (experimental only—do not use) .....	106
flex.reservation.query (experimental only—do not use) .....	106
flex.reservation.modify (experimental only—do not use) .....	107
flex.reservation.status (experimental only—do not use) .....	107
flex.reservation.enumerate (experimental only—do not use) .....	108
flex.reservation.deletions.enumerate (experimental only—do not use) .....	108
flex.resource.query .....	108
flex.resource.status .....	110
logs.syslog.modify .....	111
logs.syslog.query .....	111
services.modify .....	111
services.query .....	112
sip.modify .....	113
sip.query .....	114
system.info .....	114
user.create .....	116
user.destroy .....	116
user.modify .....	117
user.enumerate .....	117
Related Information .....	118
system.xml on 8710 and 7010 .....	118
system.xml on Media 310/320 .....	120
system.xml on Virtual Machine .....	121
Fault Codes .....	122
Example XML-RPC Response to flex.conference.create .....	125
Remotely Managed API Change History .....	127
<b>Cisco Legal Information .....</b>	<b>138</b>
<b>Cisco Trademark .....</b>	<b>138</b>



## Overview

This guide describes the API available in version 4.3 of Cisco TelePresence Server. Version 4.3 or later only supports flexible operation mode (remotely managed).

**Flexible Operation Mode**, [page 8](#) describes the API available in 4.3. This corresponds to the *remotely managed* mode of operation as described in the user interface and online help.

**Note:** TelePresence Server 4.2 was the last release to support locally managed mode. From this release (4.3) locally managed mode is no longer supported. For information about the API available in standalone operation mode (locally managed), see the TelePresence Server 4.2 API Guide at <http://www.cisco.com/c/en/us/support/conferencing/telepresence-server/products-programming-reference-guides-list.html>.

The `operationMode` parameter of the `system.info` method returns the current operation mode.



## Flexible Operation Mode

This section describes the API available in flexible operation mode (remotely managed).

**Note:** TelePresence Server 4.2 was the last release to support Locally managed mode. From this release, 4.3 onwards Locally managed mode is no longer supported. For information about the API available in standalone operation mode (locally managed), refer to the TelePresence Server 4.2 API Guide at <http://www.cisco.com/c/en/us/support/conferencing/telepresence-server/products-programming-reference-guides-list.html>.

## Introduction

This document accompanies the latest version of the management API for the Cisco TelePresence Server software when running in flexible (remotely managed) mode. The following Cisco TelePresence products support this API when they are running TelePresence Server version 4.3 and later:

- Cisco TelePresence Server MSE 8710
- Cisco TelePresence Server 7010
- Cisco TelePresence Server on Multiparty Media 310/320
- Cisco TelePresence Server on Virtual Machine
- Cisco Multiparty Media 400v
- Cisco Multiparty Media 410v
- Cisco TelePresence Server on Multiparty Media 820

## Remotely Managed Mode API Change Summary

The latest Cisco TelePresence Server API is version 4.3. The table below contains a summary of the latest changes to the remotely managed mode API. For changes introduced in older versions, see [Remotely Managed API Change History, page 127](#).



Table 1 API version 4.3 change summary

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.conference.create, page 57</a>	customPINEntryFailedMessage useCustomPINEntryFailedMessage customDisconnectPlatitudeMessage useCustomDisconnectPlatitudeMessage customWelcomeScreenAudio useCustomWelcomeScreenAudio customPINEntryAudio useCustomPINEntryAudio customOptionalPINEntryAudio useCustomOptionalPINEntryAudio customPINIncorrectAudio useCustomPINIncorrectAudio customConferenceEndedExitAudio useCustomConferenceEndedExitAudio customParticipantDisconnectedExitAudio useCustomParticipantDisconnectedExitAudio customPINFailedExitAudio useCustomPINFailedExitAudio customWaitingForChairAudio useCustomWaitingForChairAudio customOnlyParticipantAudio useCustomOnlyParticipantAudio resourceOptimizationMode customBackgroundImageURL	New parameters
	useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage	Removed
<a href="#">flex.conference.modify, page 67</a>	customPINEntryFailedMessage useCustomPINEntryFailedMessage customDisconnectPlatitudeMessage useCustomDisconnectPlatitudeMessage	New parameters
	useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage	Removed

Table 1 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.conference.query, page 72</a>	customPINEntryFailedMessage useCustomPINEntryFailedMessage customDisconnectPlatitudeMessage useCustomDisconnectPlatitudeMessage customWelcomeScreenAudio useCustomWelcomeScreenAudio customPINEntryAudio useCustomPINEntryAudio customOptionalPINEntryAudio useCustomOptionalPINEntryAudio customPINIncorrectAudio useCustomPINIncorrectAudio customConferenceEndedExitAudio useCustomConferenceEndedExitAudio customParticipantDisconnectedExitAudio useCustomParticipantDisconnectedExitAudio customPINFailedExitAudio useCustomPINFailedExitAudio customWaitingForChairAudio useCustomWaitingForChairAudio customOnlyParticipantAudio useCustomOnlyParticipantAudio resourceOptimizationMode customBackgroundImageURL	New parameters
	useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage	Removed
<a href="#">flex.conference.sendUserMessage, page 78</a>	message	Limited to 100 text characters
<a href="#">flex.conference.sendUserMessage, page 78</a>	position	Removed
<a href="#">flex.participant.sendUserMessage, page 102</a>	message	Limited to 100 text characters
<a href="#">flex.participant.sendUserMessage, page 102</a>	position	Removed
<a href="#">flex.conference.status, page 78</a>	conferenceCreditsAllocated conferenceCreditsRequired conferenceCreditsUsed	New parameters (experimental only—do not use)
<a href="#">flex.resource.status, page 110</a>	conferenceCreditsAllocated conferenceCreditsRequired conferenceCreditsUsed	New parameters (experimental only—do not use)

Table 1 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.conference.enumerate</a> , page 64	conferenceCreditsAllocated conferenceCreditsRequired conferenceCreditsUsed	New parameters (experimental only—do not use)
Table 28 Participant role enumerated type (experimental only—do not use), page 32	any accessLevelChair accessLevelGuest cascade	New enumerated type (experimental only—do not use)
Table 29 Resource optimization mode enumerated type, page 32	byParticipant byConference (experimental only—do not use)	New enumerated type
Enumerated Reservation Information Struct (experimental only—do not use), page 41	reservationID reservationReference conferenceID participantRole numParticipants maxQuality totalReservedCredits numParticipantsRemaining numCreditsRemaining	New struct (experimental only—do not use)
<a href="#">flex.reservation.create</a> (experimental only—do not use), page 106	reservationReference conferenceID participantRole numParticipants maxQuality reservationId reservationReference	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.query</a> (experimental only—do not use), page 106	reservationId reservationReference conferenceID participantRole numParticipants maxQuality totalReservedCredits	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.modify</a> (experimental only—do not use), page 107	reservationId reservationReference numParticipants maxQuality	New command and parameters (experimental only—do not use)

**Table 1 API version 4.3 change summary (continued)**

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.reservation.status (experimental only—do not use)</a> , page 107	reservationID reservationReference numParticipantsRemaining numCreditsRemaining	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.destroy (experimental only—do not use)</a> , page 1	reservationId	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.enumerate (experimental only—do not use)</a> , page 108	cookie max conferenceID moreAvailable reservations	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.deletions.enumerate (experimental only—do not use)</a> , page 108	cookie max conferenceID moreAvailable IDs	New command and parameters (experimental only—do not use)

## Design Considerations

Every API command that your application sends incurs a processing overhead within the device's own application. The amount of the overhead varies widely with the type of command and the parameters sent. If the device receives a high number of API commands every second, its performance could be seriously impaired (in the same way as if multiple users simultaneously accessed it via the web interface).

### Minimizing API Overhead

It is essential to design your application architecture and software so that the processing load on the device application is minimized.

To do this we recommend that you do the following:

- Use a single server to run the API application and to send commands to the device.
- If multiple users need to use the application simultaneously, provide a web interface on that server or write a client that communicates with the server. Then use the server to manage the clients' requests and send API commands directly to the device.
- Implement some form of control in the API application on your server to prevent the device being overloaded with API requests.

These measures provide much more control than having the clients send API commands directly, and will prevent the device performance being impaired by unmanageable numbers of API requests.

## Unavailable or Irrelevant Data

The API is designed to minimize impact on the network when responding to requests, and device responses do not routinely include either irrelevant data or empty data structures where the data is unavailable.

It follows that your application should take responsibility for checking whether a response includes the expected data, and should be designed for graceful handling of situations where the device does not respond with the expected data.

## XML-RPC Implementation

The API is implemented as messages sent using the XML-RPC protocol. This is a simple protocol for remote procedure calling that uses HTTP (or HTTPS) as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible while allowing for complex data structures to be transmitted, processed and returned. It has no platform or software dependence and was chosen in favor of SOAP (Simple Object Access Protocol) because of its simplicity.

The API implements all parameters and returned data as `<struct>` elements, each of which is explicitly named. For example, the `device.query` call returns the current time as a structure member named `currentTime` rather than as a single `<dateTime.iso8601>` value:

```
<member>
  <name>
    currentTime
  </name>
  <value>
    <dateTime.iso8601>
      20130218T10:45:00
    </dateTime.iso8601>
  </value>
</member>
```

Refer to the [XML-RPC specification](#) for more information.

## Transport

The device implements HTTP/1.1 as defined by [RFC 2616](#). It expects to receive communications over TCP/IP connections to port 80 (default HTTP port) or port 443 (default HTTPS port).

Your application should send HTTP POST messages to the URL defined by path `/RPC2` on the device's IP address, for example `https://10.0.0.53/RPC2`.

You can configure the device to receive HTTP and HTTPS on non-standard TCP port numbers if necessary, in which case append the non-standard port number to the IP address.

## Encoding

Your application should encode messages as UTF-8.

## Message Flow

The application initiates the communication and sends a correctly formatted XML-RPC command to the device.

## Example Command

```
<?xml version='1.0' encoding='UTF-8'?>
  <methodCall>
    <methodName>flex.conference.destroy</methodName>
    <params>
      <param>
```

```

<value>
  <struct>
    <member>
      <name>authenticationPassword</name>
      <value><string></string></value>
    </member>
    <member>
      <name>conferenceID</name>
      <value><string>6f030fa0-08c4-11e2-a57e-000d07100000</string></value>
    </member>
    <member>
      <name>authenticationUser</name>
      <value><string>admin</string></value>
    </member>
  </struct>
</value>
</param>
</params>
</methodCall>

```

Assuming the command was well formed and that the device is responsive, the device will respond in one of these ways:

- If the command was successful:
  - If the API method returns parameters, the device responds with an XML `<methodResponse>` message containing a structure of return parameters, as documented for each command in the [Flexible API Command Reference, page 41](#).
  - If the API method does not return parameters, the device responds with an XML `<methodResponse>` message containing a structure consisting of the single element `status` with value `operation successful`.
- If the command was unsuccessful, the device responds with an XML `<methodResponse>` that includes only a `<fault>` structure. See [Fault Codes, page 122](#).

## Example Success Response where the API Method Does Not Return Parameters

```

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value>
              <string>operation successful</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>

```

## Example Fault Response

```

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>

```

```

    <value>
      <int>4</int>
    </value>
  </member>
</member>
<member>
  <name>faultString</name>
  <value>
    <string>conferenceID: no such conference</string>
  </value>
</member>
</struct>
</value>
</fault>
</methodResponse>

```

## Data Types and Sizes

**Note:** The total size of a request or response is 32 KB. If the TelePresence Server needs to truncate a response it will either provide a mechanism for you to retrieve the remaining data or return an appropriate fault code.

The Cisco TelePresence Server API accepts the following XML-RPC types. The table includes the default sizes that your application can assume unless a more specific limit is given in a parameter description.

**Table 2 API data types and sizes**

Type	Default size accepted
<string>	31 characters
<int>	Four byte signed (-2147483648 to 2147483647)
<boolean>	1 or 0, true or false
<base64>	Not explicitly limited unless otherwise stated
<dateTime.iso8601>	ISO 8601 format eg. 20140107T13:31:26
<array>	N/A
<struct>	N/A

## HTTP keep-alives

Your application can use HTTP keep-alives to reduce the amount of TCP traffic that results from constantly polling the device. Any client which supports HTTP keep-alives may include the following line in the HTTP header of an API request:

```
Connection: Keep-Alive
```

This indicates to the device that the client supports HTTP keep-alives. The device may then choose to maintain the TCP connection after it has responded. If the device will close the connection it returns the following HTTP header in its response:

```
Connection: close
```

If this line is not in the HTTP header of the response, the client may use the same connection for a subsequent request.

The device will not keep a connection alive if:

- the current connection has already serviced the allowed number of requests
- the current connection has already been open for the allowed amount of time
- the number of open connections exceeds the allowed number if this connection is maintained

These restrictions are in place to limit the resources associated with open connections. If a connection is terminated for either of the first two reasons, the client will probably find that the connection is maintained after the next request.

**Note:** The client should never assume a connection will be maintained. Also, the device will close an open connection if the client does not make any further requests within a minute. There is little benefit to keeping unused connections open for such long periods.

## API Overview

### Authentication

**Note:** Authentication information is sent using plain text and should only be sent over a trusted network.

The controlling application must authenticate itself on the device as a user with administrative privileges. Also, because the interface is stateless, every call must contain the following authentication parameters:

**Table 3 Authentication parameters**

Parameter name	Type	Description
<code>authenticationUser</code>	string	<b>Required.</b> User name.
<code>authenticationPassword</code>	string	<b>Required.</b> User password.

If the user name and password are not recognized by the TelePresence Server, the method call fails with authentication errors.

### Identifiers and Client References

Identifiers and client references are string fields up to 50 characters in length.

#### Identifiers

The TelePresence Server assigns identifiers to resources and conferencing objects (conferences, calls). Identifiers within a pool of resource or object types are unique.

API clients must use identifiers to refer to resources and conferencing objects. The format and content of the identifier strings is subject to change and clients should not rely on any characteristics of identifiers.

Identifier fields have well-defined names that are used consistently within the TelePresence Server XML-RPC schema:

- `conferenceID`: unique identifier for a conference assigned by the TelePresence Server at conference instantiation.
- `callID`: unique identifier for a call assigned by the TelePresence Server at call instantiation.
- `participantID`: unique identifier for a participant. A participant can have one or more associated calls.

#### Client References

Client references are strings associated with objects created by this API. The content of these strings is set by clients of this API.

Client references make it possible for clients to create their own associations for objects.

The TelePresence Server does not use client references for any purpose other than to return the client reference associated with an object on request.

Client reference fields have well-defined names that are used consistently within the XML-RPC schema of this API:



- `conferenceReference`: client reference for conferences.
- `participantReference`: client reference for participants.

Client reference strings are only returned if they are not empty.

## Conference URI Identifiers

The conference URI is an identifier that allows matching of incoming calls to conferences. A conference URI can take either of the following forms:

- `username@domain`
- `123-ABC_example.com`

Valid characters are as follows:

- 0 through 9
- a through z
- A through Z
- `._-@` (only one occurrence of `@` is allowed)

## URI Matching and Connection of Incoming Calls

Suppose that incoming calls dial in to an address on a TelePresence Server. The address dialed by the incoming call is matched to a URI and connected to a conference using the following algorithm:

1. Search for a URI that is an exact match for the address. If found, connect the call to the associated conference.
2. Strip the domain part of the address (if any) and search for a URI that is an exact match. If found, connect the call to the associated conference.
3. Reject the call.

## Examples of URI Matching

These examples illustrate how matching works for conference URIs with domains.

1. URI = `conference_1@example.com`
  - Call to `conference_1@example.com` will succeed
  - Call to `conference_1@tower.example.com` will fail
  - Call to `conference_1` will fail
2. URI = `123456@example.com`
  - Call to `123456@example.com` will succeed
  - Call to `123456@tower.example.com` will fail
  - Call to `123456` will fail
3. URI = `conference_1`
  - Call to `conference_1@example.com` will succeed
  - Call to `conference_1@tower.example.com` will succeed
  - Call to `conference_1` will succeed

4. URI = 123456
  - Call to 123456@example.com will succeed
  - Call to 123456@tower.example.com will succeed
  - Call to 123456 will succeed
5. Conference 1 has URI = 789. Conference 2 has URI = 789@tower.example.com. Conference 3 has URI = 789@example.com
  - Call to 789@example.com will succeed in being connected to conference 3.
  - Call to 789@tower.example.com will succeed in being connected to conference 2.
  - Call to 789 will succeed in being connected to conference 1.
6. Conference 1 has URI = 789. Conference 2 has URI = 789@example.com
  - Call to 789@example.com will succeed in being connected to conference 2.
  - Call to 789@tower.example.com will succeed in being connected to conference 1.
  - Call to 789 will succeed in being connected to conference 1.

## URIs Do Not Need to be Unique

Under certain conditions, URIs do not have to be unique, the TelePresence Server will allow the use of the same URI in multiple conference connection definitions, if the following conditions are met:

- The conference connection definitions are bound to the same conference.
- The conference connection definitions have different PINs.
- The URI is not in active use as a participant URI in any conference during the active period of the conference.
- One of the PINs (but not both) may be blank if the same URI is in use.

## Participants

A participant can be an entity connected to a conference using one or more calls. Participant connections can be any of the following:

- Single-screen, single call connection.
- Multiscreen, single call connection.
- Multiscreen, multiple call connection.

Participants are implicitly created for incoming calls connecting to conference URIs. All other participants must be explicitly created using the API.

The API supports the creation of single and multi-call participants for which the calls can be incoming or outgoing. In the case of multi-call participants, the API supports combinations of incoming and outgoing calls.

## Participant Conference URIs

A participant can have associated conference URIs that are distinct from the URIs defined for a conference. These are called participant conference URIs. Each participant conference URI supports a single active call only. Incoming calls on participant conference URIs are connected to the conference as defined by the participant.

Participant conference URIs are bound to the conference and hence the activation and lifetimes do not exceed conference activation and lifetimes.

A participant can be configured to allow further incoming calls on a participant conference URI to be rejected or to replace the existing call.

## Creating Outgoing Calls

The following rules apply for participant outgoing call creation:

- If all the participant calls are outgoing, the calls are created immediately (that is, on creation of the participant).
- If some but not all of the calls are outgoing, the outgoing calls are created after all incoming calls for the participant have connected.
- A PIN is not accepted by [flex.participant.create, page 84](#) if all the calls are outgoing, because the TelePresence Server never requests a PIN when it has dialed out to an endpoint; in this case, the TelePresence Server will return fault code 102.

## Participant Attributes

Each participant has a unique identifier (assigned by the TelePresence Server) : `participantID`, and optionally a client-supplied reference: `participantReference`. See [Identifiers and Client References, page 16](#).

After a participant has been created, only the display name, call attributes, and media resources can be modified.

A single set of call attributes is defined for a participant, which apply to all calls belonging to a participant.

Each participant has a call nominated as the content transmitter and receiver and another as the audio transmitter and receiver. In the case of single-call participants, the content and audio transmitter and receiver can only be the single call that forms the participant.

A single PIN number specification is used and this can be input on the call nominated as the audio transmitter and receiver.

The media credits and tokens configured for a participant are reserved by the TelePresence Server for use by the calls that are members of the participant. The reservation exists for the lifetime of the participant.

All methods except for [flex.participant.create, page 84](#) require the `participantID` field to identify a participant in the conference. If the `participantID` supplied is invalid, methods fail with a "no such participant" fault.

All methods that return information return the `participantID` field, and the client-supplied `participantReference` field if one was supplied.

## Participant Lifespan

A participant is associated with one and only one conference. The lifetime of a participant cannot exceed the lifetime of the conference with which it is associated. The activation time of a participant is bound to the activation time of the conference.

A participant is destroyed automatically when any call belonging to the participant hangs up. The exception to this rule applies to participants created using this API that have incoming calls: these participants persist for the duration of the conference, unless they are destroyed explicitly using this API. Also, if any one call belonging to such a participant hangs up, all other calls connected to the participant are disconnected.

If a participant is configured with `deferConnect` enabled, then it is not destroyed when its calls are disconnected. The participant remains in the conference and will be redialed when other participants join.

## Participant Media Distribution

The media resource values are distributed to calls forming the participant according to the following rules:

- Main video tokens are divided appropriately amongst all calls in the participant.
- Extended video tokens are assigned to the nominated content transmitter and receiver.
- Audio tokens are assigned to the nominated audio transmitter and receiver.
- Media credits must be sufficient for the sum of all tokens specified.

## "Unlimited" Integers

Some parameters exchanged by this API represent configuration options that can have a greater value than what can be represented by a four byte integer.

For these options, the API and the client application can exchange a boolean version of the integer parameter which is set to `true` if the integer is unlimited. The naming convention for the boolean parameter is to append `unlimited` to the name of the associated integer parameter.

When such a value is exchanged, only one of the two types may be supplied and only one will be returned. The Cisco TelePresence Server adheres strictly to this rule, and will return a fault if your application attempts to pass both.

For example, consider an integer field called `duration` with valid values  $\geq 0$ . The associated boolean field is named `durationUnlimited`.

The following table describes the XML encoding for all settings of `duration`.

**Table 4 Example of "Unlimited" integer**

Value		XML	
	<code>name</code>	<code>type</code>	<code>value</code>
0 to 2147483647	<code>duration</code>	integer	0 to 2147483647
infinity / unlimited	<code>durationUnlimited</code>	boolean	<code>true</code>

When supplying values:

- Only one of the two parameters is required
- The `boolean` is implicitly `false` if an `int` is supplied
- If the `boolean` is `true`, the `int` must not be supplied, or the Cisco TelePresence Server will return a fault

## Media Credits

Every participant that connects to a conference consumes a number of credits.

**Note:** The token requirements for a call cannot be known prior to instantiation of the call, so no checks are made on `flex.participant.create` or `flex.participant.modify` to determine if the call will have adequate resources. The client is therefore responsible for ensuring that the call has adequate resources.

The number of credits required for a given participant can be derived using the sum of the tokens (main video, extended video and audio) required for the participant and the `mediaCreditTokenRanges` array returned by [flex.resource.query](#), page 108.

The array returned is effectively a conversion table from media credits to media tokens. For example, if the array returned is [48, 315, 630, 840, 1260, 2520, 3780, 5040, 7560, 10080], this can be interpreted as the following conversion table:

**Table 5 Media credits to media tokens mapping**

Media credits	Media tokens
48	0 to 48
315	49 to 315
630	316 to 630
840	631 to 840

**Table 5 Media credits to media tokens mapping (continued)**

Media credits	Media tokens
1260	841 to 1260
2520	1261 to 2520
3780	2521 to 3780
5040	3781 to 5040
7560	5041 to 7560
10080	7561 to 10080

If media credit values supplied to API methods do not match any of the values in the "Media credits" column in the previous table, the value is rounded down to the next level. Supplying media credit values less than 48 allocates 0 credits.

Every participant must have enough credits to use the tokens configured for that participant. API methods fail if this requirement is not met. This applies to media credit values rounded down as described previously.

Participant calls are rejected if there are insufficient credits when connecting the call to the conference.

## Media Reservation

Reserved media resources are tokens and credits that have been assigned for exclusive use by a participant. Reservation guarantees that if an endpoint connection succeeds, media resources required to service the connection exist.

**Note:** The token requirements for a call cannot be known prior to instantiation of the call, so no checks are made on `flex.participant.create` or `flex.participant.modify` to determine if the call will have adequate resources. The client is therefore responsible for ensuring that the call has adequate resources.

## Enumeration

This API supports incremental enumeration of objects such as conferences and participants. The following methods are typically associated with complete enumeration of a type of object:

- `flex.object.enumerate`
- `flex.object.deletions.enumerate`

Both methods use cookies to determine what content needs to be returned. To start the enumeration, the methods should be invoked without supplying a cookie. To continue the enumeration, the methods should be invoked with the cookie returned by the previous invocation.

Both methods return the boolean parameter `moreAvailable`. If the value of this parameter is `true`, more data is available.

For information on how you can use incremental enumeration to optimize resource usage, see [flex.participant.media.enumerate, page 89](#).

## The `.enumerate` Methods

The `.enumerate` methods are intended for enumeration of live objects and return lists of `object` that are new or have been revised.

To use the `.enumerate` methods:

- On the first invocation, do not present a cookie. Information is returned on all live objects.
- On subsequent invocations, present a cookie. Information is returned on live objects that have changed or have been added since the previous invocation as indicated by the cookie.

The `.enumerate` methods may fail with `Fault 102: 'cookie is invalid or expired'` if the enumeration cannot be completed; that is, if all changes or additions that occurred since the last invocation cannot be listed. In such cases, restart both live and deletions enumerations, discarding the previous state.

## The `.deletions.enumerate` Methods

The `.deletions.enumerate` methods are intended for enumeration of objects that have been destroyed. They return lists of *object* IDs that have been deleted since the last invocation as indicated by the cookie.

To use the `.deletions.enumerate` methods:

- On the first invocation, do not present a cookie. No IDs are returned for the first invocation.
- On subsequent invocations, present a cookie. IDs of objects that have been deleted since the previous invocation of the method are returned.

The `deletions.enumerate` methods may fail with `Fault 102: 'cookie is invalid or expired'` if the enumeration cannot be completed; that is, if all deletions that occurred since the last invocation cannot be listed. In such cases, restart both live and deletions enumerations, discarding the previous state.

## Enumeration Method Invocation

Typically, the enumeration methods should be invoked in response to feedback notification of an event. If the methods are invoked and no changes have occurred (since the last invocation as determined by the cookie), empty lists will be returned.

For example, to maintain a list of information about live conferences:

1. Invoke [flex.conference.deletions.enumerate, page 64](#) with no parameters other than the [Authentication, page 16](#) parameters, and store the `cookie` string parameter returned as the *deletions cookie*.
2. Invoke [flex.conference.enumerate, page 64](#) with no parameters other than authentication parameters.
3. Go to step 5.
4. Invoke [flex.conference.enumerate, page 64](#) with the `cookie` parameter set to the *live objects cookie*.
5. Store the `cookie` string parameter returned as the *live objects cookie*.
6. Process the list of conferences returned.
7. If `moreAvailable` is `true`, repeat from step 4.
8. Go to step 13.
9. Invoke [flex.conference.deletions.enumerate, page 64](#) with the `cookie` parameter set to the *deletions cookie* (see above and also below).
10. Store the `cookie` parameter returned as the *deletions cookie*.
11. Process the `conferenceIDs` array.
12. If `moreAvailable` is `true`, repeat from step 9.
13. Wait for feedback notification.
14. In the event of a conference change or addition, go to step 4.
15. In the event of a conference deletion, go to step 9.

In the algorithm above, at the start of the enumeration, [flex.conference.deletions.enumerate, page 64](#) is invoked before [flex.conference.enumerate, page 64](#). This ensures that the deletion of any conference returned by [flex.conference.enumerate, page 64](#) will be returned by [flex.conference.deletions.enumerate, page 64](#) when it occurs.

It is also possible that [flex.conference.deletions.enumerate, page 64](#) will return the IDs of conferences which have not been returned by [flex.conference.enumerate, page 64](#). This can happen when a conference is created and destroyed before [flex.conference.enumerate, page 64](#) is invoked or the enumeration has not proceeded far enough to return the conference ID.

Participants and participant media resources can be enumerated in a similar way using the following methods:

- [flex.participant.enumerate, page 87](#) and/or [flex.participant.media.enumerate, page 89](#)
- [flex.participant.deletions.enumerate, page 86](#)

## Feedback Events and Enumeration

Feedback events are generated to aid incremental enumeration of conferences and participants. See [Feedback Events, page 24](#) for more information.

For conferences:

- When a conference is created, modified or its state changes, the `flexConferenceEnum` event is generated. Invoke [flex.conference.enumerate, page 64](#) to retrieve information for newly added or modified conferences. The `flexConferenceEnum` event is only generated for those modifications or state changes that affect data returned by [flex.conference.enumerate, page 64](#).
- When a conference is destroyed, the `flexConferenceDeletionsEnum` event is generated. Invoke [flex.conference.deletions.enumerate, page 64](#) to identify which conferences have been destroyed.

Similarly for participants:

- When a participant is created, modified or its state changes, the `flexParticipantEnum` event is generated. Invoke [flex.participant.enumerate, page 87](#) to retrieve information for newly added or modified participants. The `flexParticipantEnum` event is only generated for those modifications or state changes that affect the data returned by the [flex.participant.enumerate, page 87](#) method.
- When a participant is created or its media resource state changes, the `flexParticipantMediaEnum` event is generated. Invoke [flex.participant.media.enumerate, page 89](#) to retrieve participant media information. The `flexParticipantMediaEnum` event is only generated for those modifications or state changes that affect the data returned by the [flex.participant.media.enumerate, page 89](#) method.
- When a participant is destroyed, the `flexParticipantDeletionsEnum` event is generated. Invoke [flex.participant.deletions.enumerate, page 86](#) to identify which participants have been destroyed.

## DTMF

The set of valid characters for DTMF is:

- `*#0123456789ABCD,`

The comma character is used to insert delay. Each comma denotes a two-second delay.

Commands that take DTMF string parameters will accept any non-DTMF ASCII characters in the string but the TelePresence Server will ignore them; it processes the string until it reaches the end, sending only the tones for characters within the set `*#0123456789ABCD` and pausing the tone sequence by two seconds for each comma.

The TelePresence Server returns a fault if there are non-ASCII characters in the string.

## Feedback Receivers

The API allows you to register your application as a feedback receiver. This means that the application does not have to constantly poll the device if it wants to monitor activity. By using feedback events, you can avoid imposing the high loads that polling can cause especially when there are multiple API users.

The device publishes events when they occur. If the device knows that your application is listening for these events, it will send XML-RPC messages to your application's interface when the events occur.

**Note:** The TelePresence Server expects your application to provide at least an HTTP 200 OK status header. The TelePresence Server logs a warning event if it cannot be sure your application received the feedback message.

- Use [feedbackReceiver.configure](#), page 53 to register a receiver to listen for one or more [Feedback Events](#), page 24.
- Use [feedbackReceiver.query](#), page 54 to return a list of receivers that are configured on the device.
- Use [feedbackReceiver.reconfigure](#), page 55 to change the configuration of an existing feedback receiver.
- Use [feedbackReceiver.remove](#), page 55 to remove an existing feedback receiver.
- Use [feedbackReceiver.status](#), page 55 to display the status of a specific feedback receiver, and all the events to which it is subscribed.

After registering as a feedback receiver, the application will receive feedback messages on the specified interface.

## Feedback Messages

The feedback messages follow the format used by the device for XML-RPC responses.

The messages contain two parameters:

- `sourceIdentifier` is a string that identifies the device, which may have been set by `feedbackReceiver.configure` or `feedbackReceiver.reconfigure`. If it has not been set it will be the device's MAC address.
- `events` is an array of strings that contain the names of the feedback events that have occurred.

### Example feedback message

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
  <methodName>eventNotification</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>sourceIdentifier</name>
            <value><string>000D7C000C66</string></value>
          </member>
          <member>
            <name>events</name>
            <value>
              <array>
                <data>
                  <value><string>restart</string></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

## Feedback Events

The following table lists the feedback events that the TelePresence Server can publish:



**Table 6 Feedback events**

Event	Description
cdrAdded	One or more new Call Detail Records have been logged
configureAck	The source publishes this event to acknowledge that an application has successfully added, reconfigured, or removed a feedback receiver
deviceStatusChanged	Generated when the TelePresence Server is shut down or a feature key is added or removed. Invoke <code>device.query</code> for more details.
flexAlive	Alive notification. This event is generated every 10 seconds and should be used if it is a requirement to monitor whether the TelePresence Server is alive. See <a href="#">When to consider a TelePresence Server to be no longer alive</a> .
flexConferenceDeletionsEnum	One or more conferences have been destroyed. Invoke <a href="#">flex.conference.deletions.enumerate, page 64</a> to get the identifiers of conferences destroyed.
flexConferenceEnum	The state of one or more conferences has changed. Invoke <a href="#">flex.conference.enumerate, page 64</a> to get the changes.
flexResourceConfiguration	The resource configuration has changed. Media blades have been added or removed. Invoke <a href="#">flex.resource.query, page 108</a> to retrieve the new configuration.
flexParticipantAdvancedEnum	Participants have been created or their state has changed. Invoke <a href="#">flex.participant.advanced.enumerate, page 80</a> to get the changes.
flexParticipantDeletionsEnum	Participants have been destroyed. Invoke <a href="#">flex.participant.deletions.enumerate, page 86</a> to get the identifiers of participants destroyed.
flexParticipantEnum	Participants have been created or their state has changed. Invoke <a href="#">flex.participant.enumerate, page 87</a> to get the changes.
flexParticipantMediaEnum	Participants have been created or their media resources state has changed. Invoke <a href="#">flex.participant.media.enumerate, page 89</a> to get the changes.
flexResourceStatus	Resource usage has changed: calls, participants, conferences, media tokens, and media credits. Invoke <a href="#">flex.resource.status, page 110</a> to get the changes.
receiverDeleted	The feedback receiver receiving this event has been stopped and its configuration deleted or the URI of the feedback receiver has been changed, in which case this event is sent to the previous URI.
receiverModified	The feedback receiver receiving this event has been modified.
restart	The TelePresence Server has restarted or booted.

**Note:** When the URI of a feedback receiver is changed, `receiverModified` and `receiverDeleted` events are sent to the previous URI of the feedback receiver.

When to consider a TelePresence Server to be no longer alive

When monitoring the liveness of a TelePresence Server, it should be considered alive if the time since the last `flexAlive` feedback event does not exceed twice the feedback interval (that is, 20 seconds). After this time, the status of the server should be checked with `flex.resource.status`, [page 110](#); only if there is no response should the server be considered no longer alive.

## Data structures and enumerated types

### Enumerated Types

Enumerated types as described here are a convenient way of describing the behavior of string fields for which arbitrary string values are not appropriate. Enumerated types are not an extension to the XML-RPC specification.

Each enumerated type has an associated list of strings. If a parameter's value is described as belonging to a particular enumerated type:

- Input strings that are not in the list will generate an invalid parameter fault.
- Only strings that are in the list will be returned by the TelePresence Server.
- The maximum length of the returned string is the same as the length of the longest string in the list.

#### Enumerated types described in this topic:

- [Table 7 Access level enumerated type, page 27](#)
- [Table 8 Motion vs Sharpness enumerated type, page 27](#)
- [Table 9 Aspect ratio enumerated type, page 27](#)
- [Table 10 Single screen layout enumerated type, page 27](#)
- [Table 11 Multi-screen layout enumerated type, page 27](#)
- [Table 12 Call protocol enumerated type, page 27](#)
- [Table 13 Video format enumerated type, page 28](#)
- [Table 14 Participant encryption enumerated type, page 28](#)
- [Table 15 Video transmit size enumerated type, page 28](#)
- [Table 16 Call conference state enumerated type, page 28](#)
- [Table 17 Call state enumerated type, page 28](#)
- [Table 18 Cascade roles enumerated type, page 29](#)
- [Table 19 Optimization profiles enumerated type, page 29](#)
- [Table 20 Switching mode enumerated type, page 29](#)
- [Table 21 Multistream mode enumerated type, page 30](#)
- [Table 22 Participant connection state enumerated type, page 30](#)
- [Table 23 Encryption status enumerated type, page 30](#)
- [Table 24 Media Status enumerated type, page 31](#)
- [Table 25 Full screen mode enumerated type, page 31](#)
- [Table 26 Audio gain mode enumerated type, page 31](#)
- [Table 27 Control level enumerated type, page 31](#)
- [Table 28 Participant role enumerated type \(experimental only—do not use\), page 32](#)
- [Table 29 Resource optimization mode enumerated type, page 32](#)

**Table 7 Access level enumerated type**

Name	Description
chair	The participant is granted chair access to the conference.
guest	The participant is granted guest access to the conference.
unknown	Incoming calls are reported as "unknown" until they are authorized (reach the lobby screen).

**Table 8 Motion vs Sharpness enumerated type**

Name	Description
favorMotion	Use high frame rates.
favorSharpness	Use high resolution.
balanced	Frame rate >= 12 fps.

**Table 9 Aspect ratio enumerated type**

Name	Description
onlyFourToThree	4:3 only.
onlySixteenToNine	16:9 only.
allowAllResolutions	Allow 4:3 as well as 16:9.

**Table 10 Single screen layout enumerated type**

Name	Description
layoutSingle	Full screen only.
layoutActivePresence	Active presence: full screen with pips.
layoutProminent	Single large pane and up to four small panes.
layoutEqual	Multiple panes of the same size.

**Table 11 Multi-screen layout enumerated type**

Name	Description
layoutSingle	Full screen only.
layoutActivePresence	Active presence: full screen with pips.

**Table 12 Call protocol enumerated type**

Name	Description
h323	H.323 Protocol.
sip	SIP (Session Initiation Protocol).

**Table 13 Video format enumerated type**

Name	Description
<code>NTSC</code>	National Television System Committee (NTSC) video format, fractions of 30 fps.
<code>PAL</code>	Phase Alternating Line (PAL) video format, fractions of 25 fps.

**Table 14 Participant encryption enumerated type**

Name	Description
<code>forbidden</code>	Encryption is denied.
<code>required</code>	Encryption is mandatory.
<code>optional</code>	Encryption is supported but not mandatory.

**Table 15 Video transmit size enumerated type**

Name	Description
<code>none</code>	Do not allow changes in video size during transmission.
<code>dynamicResolution</code>	Allow video size to be optimized during transmission.
<code>dynamicCodecAndResolution</code>	Allow video size to be optimized during transmission and/or dynamic codec selection.

**Table 16 Call conference state enumerated type**

Name	Description
<code>idle</code>	Initial state.
<code>pinEntry</code>	Call needs to enter PIN to progress.
<code>blank</code>	Showing blank video (and silent audio).
<code>welcome</code>	Showing conference welcome screen.
<code>awaitingChair</code>	Awaiting presence of one or more chair participants before activating.
<code>awaitingAccept</code>	Awaiting a chair participant to accept or deny the participation of the call in the conference.
<code>complete</code>	The call is now fully connected to the conference.
<code>disconnecting</code>	Showing exit lobby.

**Table 17 Call state enumerated type**

Name	Description
<code>callStateIdle</code>	Call is inactive.
<code>callStateAlerting</code>	Endpoint is ringing.
<code>callStateAnswering</code>	Call is in the process of being connected.
<code>callStateConnected</code>	Endpoint is connected.

**Table 17 Call state enumerated type (continued)**

Name	Description
<code>callStateRetrying</code>	Call is being retried.
<code>callStateFailed</code>	Call has failed all retries  <b>Note:</b> If the audio receiver call fails all retries in a participant with all outgoing calls then all participant calls are destroyed, so there is no longer a call state.

**Note:** For a participant with multiple calls, all of which are outgoing, then as soon as the audio receiver call fails its last retry, all participant calls are destroyed (by design). In this case it is not possible to see the call state for any of these calls.

**Table 18 Cascade roles enumerated type**

<code>cascadeRole</code> value	Description
<code>cascadeNone</code>	The participant is not a cascade link.
<code>cascadeMaster</code>	The participant is the master in a cascade link with at least one other TelePresence Server. This end is the central node in a star topology.
<code>cascadeSlave</code>	This participant is the slave in a cascade link to a master TelePresence Server. This end is the outer node in a star topology.

**Table 19 Optimization profiles enumerated type**

<code>optimizationProfile</code> value	Description
<code>maximizeEfficiency</code>	Screen licenses are conserved aggressively. This value gives the most calls for the available resources.
<code>favorEfficiency</code>	This is a balance of efficiency and experience that favors conserving screen licenses over attempting to grant the requested resolution.
<code>favorExperience</code>	Default. This is a balance of efficiency and experience that favors granting the requested resolution over conserving screen licenses.
<code>maximizeExperience</code>	Screen licenses are more readily allocated. This value gives the best experience of the four profiles.  If you disable the optimization by bandwidth (by setting <code>optimizationProfile</code> to <code>capabilitySetOnly</code> ), calls will be capable of higher resolutions at lower bandwidths but the inefficiency in allocation could well outweigh the benefit.
<code>capabilitySetOnly</code>	This is the behavior of TelePresence Server 3.1.  The TelePresence Server only considers the endpoint's maximum advertized resolution when reporting its screen license requirement to the managing system; it does not attempt to report resources based on the endpoint's advertized receive bandwidth.

**Table 20 Switching mode enumerated type**

<code>displayLayoutSwitchingMode</code> value	Description
---	-------------

**Table 20 Switching mode enumerated type (continued)**

<code>switchingRoomSwitched</code>	Room switching is used. When a participant from a room speaks, all segments (panels) from the room are shown on three-screen endpoints.
<code>switchingSegmentSwitched</code>	Segment switching is used. When a participant from a room speaks, only their segment(panel) of the room is shown on three-screen endpoints.

**Table 21 Multistream mode enumerated type**

<code>multistream</code> value	Description
<code>multistreamOff</code>	Multistream not supported.
<code>multistreamOn</code>	Multistream supported.

**Table 22 Participant connection state enumerated type**

<code>connectionState</code> value	Description
<code>disconnected</code>	All calls for this participant are disconnected
<code>connecting</code>	At least one of the participant's calls has not yet fully connected to the conference. This includes calls at the welcome screen or the "Waiting for conference chair to join" lobby screen.
<code>connected</code>	All the participant's calls are fully connected and active in the conference.
<code>onHold</code>	The participant's calls are all connected, but the participant is on hold (not sending or receiving media)
<code>disconnecting</code>	The participant is currently being shown the exit lobby before it is disconnected
<code>neverConnected</code>	Participant has at least one configured URI and has never received any calls
<code>deferred</code>	Participant has <code>deferConnect</code> enabled and is waiting for other calls
<code>partiallyFailed</code>	One or more outgoing calls have failed all retry attempts
<code>audioReceiverFailed</code>	The outgoing audio receiver call has failed on a participant that only has outgoing calls  <b>Note:</b> This may indicate that retries on all calls have failed but may not. This is because the audio receiver call failing will cause all calls to be disconnected on a participant that only has outgoing calls.

**Note:** connected means fully in conference—past lobby/pin entry.

**Table 23 Encryption status enumerated type**

<code>encryptionStatus</code> value	Description
<code>unknown</code>	The encryption status is not yet known because it is still being negotiated
<code>encrypted</code>	All of the associated channels are encrypted
<code>mixed</code>	Some of the associated channels are encrypted, others are unencrypted
<code>unencrypted</code>	All of the associated channels are unencrypted

**Table 24 Media Status enumerated type**

encryptionStatus value	Description
notNegotiated	The associated channel has not been negotiated. If there are multiple channels, this status indicates that no channels in the associated category have been negotiated.
inUse	The associated channel is in use, or at least one channel in the category is in use.
notInUse	The associated channel has been negotiated but is not being used. If there are multiple channels, this status indicates that none of them are in use.
muted	The associated channel has been negotiated but is muted. If there are multiple channels, this status indicates that all of the channels in the associated category have been muted.

**Table 25 Full screen mode enumerated type**

Name	Description
never	The stream from the single-screen participant will never show in a full-screen pane when viewed on a multiscreen endpoint.
always	The stream from the single-screen participant is always allowed to show in a full-screen pane on a multi-screen endpoint.
dynamic	The stream from the single-screen participant is allowed to show in a full-screen pane on a multiscreen endpoint, as for <code>allowed</code> .  However, if there are other multiscreen endpoints to show, the single-screen participant will not show in a full-screen pane on a multiscreen endpoint. In this case, the view of the single-screen endpoint will be restricted to a smaller, continuous presence pane.

**Table 26 Audio gain mode enumerated type**

Name	Description
gainModeDisabled	No gain is applied to the audio.
gainModeAutomatic	The level of gain applied is calculated automatically to normalise audio levels in a conference.
gainModeFixed	The level of gain applied is a fixed value, supplied separately.

**Table 27 Control level enumerated type**

Name	Description
controlNone	The participant is not authorized to control any conference settings.
controlLocal	The participant is authorized to control local conference settings—i.e. those related to the participant's own endpoint—but is not authorized to control the conference settings that affect other participants.
controlConference	The participant is authorized to control conference settings that may affect other participants, such as locking the conference or disconnecting other participants, and is also authorized to control the local conference settings (such as changing the conference layout shown on the local endpoint).

**Table 28 Participant role enumerated type (experimental only—do not use)**

Name	Description
<code>any</code>	Any participant.
<code>accessLevelChair</code>	A participant with an <code>accessLevel</code> of <code>chair</code> .
<code>accessLevelGuest</code>	A participant with an <code>accessLevel</code> of <code>guest</code> .
<code>cascade</code>	A participant with a <code>cascadeRole</code> of <code>cascadeMaster</code> or <code>cascadeSlave</code> .

This enumerated type is only used for reservations in `byConference` mode.

**Table 29 Resource optimization mode enumerated type**

Name	Description
<code>byParticipant</code>	Resources are handled by participant. Default.
<code>byConference</code>	Resources are aggregated across conferences (experimental only—do not use).

## Structs

The following structs are used by multiple methods in this API. Other structs that are applicable to one command alone are described with the command in the [Flexible API Command Reference, page 41](#).

- [Media Tokens Struct, page 32](#)
- [Participant Media Resources Struct, page 33](#)
- [Call Attributes Struct, page 34](#)
- [Conference URI Details Struct, page 39](#)
- [Participant Call Definition Struct, page 40](#)
- [Enumerated Reservation Information Struct \(experimental only—do not use\), page 41](#)

## Media Tokens Struct

Media tokens are used to describe how media resources should be used when assigned to conferences and participants.

Media token parameters are passed as a structure of the form described in the following tables. This struct is referred to as the `mediaTokens` struct in this document.

**Table 30 Media tokens struct members**

Parameter name	Type	Description
<code>total</code>	integer ( $\geq 0$ )	Required. Maximum total media token usage permitted across all channels.
<code>maxPerChannel</code>	integer ( $\geq 0$ , $\leq$ <code>total</code> )	Maximum media resource usage permitted for a channel. Default: see <code>maxPerChannelUnlimited</code> . Note: <code>maxPerChannel</code> must be less than <code>total</code> .
<code>maxPerChannelUnlimited</code>	boolean	Whether an unlimited number of resources can be used for a channel. See " <a href="#">Unlimited</a> " Integers, page 20. Default: <code>true</code> .



**When supplying this struct to the TelePresence Server:**

- Negative values for `maxPerChannel` and `total` are invalid parameter values.
- If neither `maxPerChannel` nor `maxPerChannelUnlimited` is specified, `maxPerChannelUnlimited` is defaulted to `true`.
- If `maxPerChannelUnlimited` is `true`, the system media tokens per channel limit applies.
- The system media tokens per channel limit can be retrieved using the [flex.resource.query, page 108](#) method and is the value of the `maxMediaTokensPerChannel` field.

**When this struct is returned by the TelePresence Server:**

- The field `maxPerChannelUnlimited` is only present if its value is set to `true`.
- The field `maxPerChannel` is only present when there is an upper limit on the number of resources that can be used and the value of `maxPerChannelUnlimited` is `false`.

## Participant Media Resources Struct

The collection of parameters in the participant media resources struct describe the media resource configuration for a participant.

This struct is referred to as the `participantMediaResources` struct in this document.

**Table 31 Participant media resources struct members**

Parameter name	Type	Description
<code>mediaTokensMainVideo</code>	struct	Media token values representing the maximum resources that can be assigned to main video within a participant. See <a href="#">Media Tokens Struct, page 32</a> .
<code>mediaTokensExtendedVideo</code>	struct	Media token values representing the maximum resources that can be assigned to extended video within a participant. See <a href="#">Media Tokens Struct, page 32</a> .
<code>mediaTokensAudio</code>	struct	Media token values representing the maximum resources that can be assigned to audio within a participant. See <a href="#">Media Tokens Struct, page 32</a> .
<code>numMediaCredits</code>	integer ( $\geq 0$ )	Number of credits configured for the participant. See <a href="#">Media Credits, page 20</a> .

All members of the `participantMediaResources` struct are mandatory. In practice, this means that the following parameters must always be present:

- `participantMediaResources.mediaTokensMainVideo.total`
- `participantMediaResources.mediaTokensExtendedVideo.total`
- `participantMediaResources.mediaTokensAudio.total`
- `participantMediaResources.numMediaCredits`

When `participantMediaResources` fields are updated, all members of the struct are changed. Unspecified optional fields are set to their default values. For example, if media resource usage per channel is not specified in the update, default values are applied and it will be set to unlimited.

**Note:** `participantMediaResources` cannot be partially updated. For example, you cannot modify audio tokens without also supplying the video and extended video tokens at the same time.

**Note:** The token requirements for a call cannot be known prior to instantiation of the call, so no checks are made on `flex.participant.create` or `flex.participant.modify` to determine if the call will have adequate resources. The client is therefore responsible for ensuring that the call has adequate resources.

`participantMediaResources.numMediaCredits` must be greater than or equal to the sum of media token values and is subject to the restrictions described in [Media Credits, page 20](#).

`participantMediaResources` can be configured at multiple points in the API. The media resources for a participant are selected in the following order of preference:

1. Participant specification, if defined (participant creation and participant modification).
2. Conference URIs, if defined.
3. Conference default specification (always defined).

## Call Attributes Struct

Where call attributes are accepted:

Call attributes can be specified in the following places, by supplying a `callAttributes` struct as described in [Table 32 callAttributes struct members, page 35](#):

- Participant specification (see [flex.participant.create, page 84](#) command).
- Participant modification (see [flex.participant.modify, page 91](#) command).
- Conference URI specification (see [Conference URI Details Struct, page 39](#)).
- Conference creation (see [flex.conference.create, page 57](#) command).
- Conference modification (see [flex.conference.modify, page 67](#) command).

## How Call Attributes are Derived

On conference creation you may supply a `callAttributes` struct, whose values then become the default for any conference URIs or participants that you subsequently create. If you don't supply the struct at that point, or omit any of its optional members, then the omitted parameters will take on the default values (as listed in [Table 32 callAttributes struct members, page 35](#)) whenever they are subsequently used or returned by the TelePresence Server.

You may also supply `callAttributes` when you create or modify conference URIs or participants. Whenever there is overlap between call attributes, the attributes specified 'nearest' the participant will take precedence, as follows:

- Call attributes supplied for a participant will take precedence over those supplied for a conference URI or a conference
- Call attributes specified for a conference URI will take precedence over those specified for the conference
- Wherever you supply an attribute it will take precedence over the default

The values of all `callAttributes` fields are set when a participant is instantiated. This means that subsequent changes to the conference's call attributes or conference URI's call attributes have no effect on existing participants.

Currently up to two call attribute instances are used as sources

- Outgoing calls
  1. Call attributes specified in `flex.participant.create` method
  2. Conference default call attributes
- Incoming calls
  - On conference URI
    1. Call attributes specified for a conference URI
      - If multiple PINs are specified for the same URI then call attributes are initially applied from the conference connection definition found using the below criteria:
        1. The conference connection definition with the highest media credits value.
        2. The conference connection definition with `accessLevel` chair.
        3. The first conference connection definition specified in the URIs array.

- When a PIN is entered, call attributes are modified to match those of the conference connection definition for the PIN entered.
2. Conference default call attributes
    - On participant conference URI
  1. PIN-specific call attributes (specified in flex.participant.create method, 'PINs' array)
    - These call attributes are not applied immediately. When a call joins the participant and correctly enters a PIN then these call attributes override the call attributes previously set by the levels below.
  2. Participant call attributes (specified in flex.participant.create method)
  3. Conference default call attributes

The following table lists the parameters that are accepted by this struct.

**Table 32** callAttributes struct members

Parameter name	Type	Description
accessLevel	string	Access level associated with this participant, when connected. See <a href="#">Table 7 Access level enumerated type, page 27</a> . Default: <code>chair</code> .  <b>Note:</b> Incoming calls are reported as <code>unknown</code> until they have been authorized (reach the lobby screen).
encryption	string	Encryption setting. See <a href="#">Table 14 Participant encryption enumerated type, page 28</a> . Default: <code>optional</code> .
autoDisconnect	boolean	Whether this call automatically disconnects if the only calls connected to a conference have <code>autoDisconnect</code> set. Default: <code>false</code> .
maxTransmitPacketSize	integer (428-1448)	Sets the maximum size (bytes) of video packets sent by the TelePresence Server, including IP headers.  If you want to set the maximum possible size for the video packets, use 1428 for an IPv4 network or 1448 for an IPv6 network.  We recommend using the default (1400), or higher, unless there is a known packet size restriction in the path. This allows the TelePresence Server to make the most efficient use of the available bandwidth.  If the packets are too large for a network that requires a smaller maximum transmission unit (MTU), network elements may fragment and reintegrate the packets which can impair performance.  <b>Note:</b> the old range is still accepted for backwards compatibility only.
packetLossThreshold	integer (0-100)	When the proportion of packets lost (per thousand packets expected) exceeds this threshold over a short period of time, packet loss is reported. Thresholds from 0 to 10% can be configured.  Default: 5 (equal to 0.5%).

Table 32 callAttributes struct members (continued)

Parameter name	Type	Description
<code>videoRxFlowControlOnErrors</code>	boolean	Flow control on video errors. Default: <code>true</code> .
<code>videoRxFlowControlOnViewedSize</code>	boolean	Flow control based on viewed size. Default: <code>true</code> .  This parameter is modified in version 4.0 as follows: if <code>true</code> , flow control is applied only when the video stream is not viewed by anyone. Formerly, this optimization would also apply when the stream was viewed at small resolutions.
<code>videoTxSizeOptimization</code>	string	Video transmit size setting. See <a href="#">Table 15 Video transmit size enumerated type, page 28</a> . Default: <code>dynamicCodecAndResolution</code> .
<code>presentationContributionAllowed</code>	boolean	Whether the endpoint can contribute presentations to a conference. Default: <code>true</code> .
<code>presentationTakeoverAllowed</code>	boolean	Whether the endpoint can start contributing presentations even if the conference already has an active presentation. Default: <code>true</code> .
<code>videoTxPresentationAllowed</code>	boolean	Whether the endpoint can receive presentations in its extended video channel (if available). Default: <code>true</code> .
<code>videoTxPresentationMainVideoAllowed</code>	boolean	Whether the endpoint can receive presentations in its main video channel (if an extended video channel is unavailable, disabled, or there are insufficient resources available). For multi-call participants, this option is always <code>false</code> . Setting it to <code>true</code> will result in it implicitly being set to <code>false</code> . Any implicit changes will be reflected in the return values of <a href="#">flex.participant.query, page 91</a> . Default: <code>true</code> .
<code>audioStereoEnabled</code>	boolean	Whether support for stereo is enabled. Default: <code>true</code> .
<code>audioDirectionalEnabled</code>	boolean	Whether directional audio is enabled. Default: <code>true</code> .
<code>indicateUnencryptedParticipants</code>	boolean	Whether the unencrypted icon is displayed. Default: <code>true</code> .
<code>indicateAudioOnlyParticipants</code>	boolean	Displays the participant overflow icon when there are more participants present in a conference than can be displayed on a given endpoint's layout. Default: <code>true</code> .
<code>mainVideoTxPictureAspectRatio</code>	string	Permissible aspect ratios for the main video channel. See <a href="#">Table 9 Aspect ratio enumerated type, page 27</a> . Default: <code>onlySixteenToNine</code> .
<code>extendedVideoTxPictureAspectRatio</code>	string	Permissible aspect ratios for the extended video channel. See <a href="#">Table 9 Aspect ratio enumerated type, page 27</a> . Default: <code>allowAllResolutions</code> .

Table 32 callAttributes struct members (continued)

Parameter name	Type	Description
videoTxFormat	string	Video format. See <a href="#">Table 13 Video format enumerated type, page 28</a> . Default: NTSC.
videoTxMotionSharpness	string	Motion sharpness setting. See <a href="#">Table 8 Motion vs Sharpness enumerated type, page 27</a> . Default: balanced.
videoRxClearVisionEnabled	boolean	Whether upscaling using ClearVision is allowed. Default: true.
video60fpsEnabled	boolean	Whether support for 60 frames per second is enabled. Default: true.
fullScreenMode	string	Setting for full screen viewing of single-screen endpoints. See <a href="#">Table 25 Full screen mode enumerated type, page 31</a> . Default: always.
displaySelfView	boolean	Whether participants see themselves as well as other participants. Default: false.
displayShowBorders	boolean	Whether panes are surrounded with a border to separate them visually. Default: true.
displayDefaultLayoutSingleScreen	string	Layout scheme used for single-screen endpoints. See <a href="#">Table 10 Single screen layout enumerated type, page 27</a> . Default: layoutActivePresence. <b>Note:</b> This has no effect on multistream calls.
displayDefaultLayoutMultiScreen	string	Layout scheme used for multiscreen endpoints. See <a href="#">Table 11 Multi-screen layout enumerated type, page 27</a> . Default: layoutActivePresence. <b>Note:</b> This has no effect on multistream calls.
displayForceDefaultLayout	boolean	If true, this prevents the participant from changing the display layout using FECC, DTMF, or ActiveControl. The layout sent to the endpoint will be forced to the configured value for that type of endpoint; that is, the value of either displayDefaultLayoutSingleScreen or displayDefaultLayoutMultiScreen. Default: false. <b>Note:</b> This has no effect on multistream calls.
displayShowEndpointNames	boolean	Whether endpoint names are shown as panel labels. Default: true.
audioReceiveGainMode	string	See <a href="#">Table 26 Audio gain mode enumerated type, page 31</a> . Either gainModeDisabled, gainModeAutomatic, or gainModeFixed. Default: gainModeAutomatic.

Table 32 callAttributes struct members (continued)

Parameter name	Type	Description
<code>audioReceiveGain</code>	integer (-12000 to +12000)	Gain for received audio when <code>audioReceiveGainMode</code> is <code>gainModeFixed</code> . The unit is millidecibels and the default value is 0.  This call attribute is not required for other values of <code>audioReceiveGainMode</code> and will be ignored if supplied in these cases.
<code>audioTransmitGain</code>	integer (-12000 to +12000)	Gain for transmitted audio, measured in millidecibels. Default: 0.
<code>forceTIP</code>	boolean	Defines whether the use of TIP (Telepresence Interoperability Protocol) is enforced, even if the endpoint does not indicate that it is TIP capable. Default: <code>false</code> .  This legacy setting is kept in the API to support legacy TIP endpoints. We do not recommend using it in any other circumstances because this could result in reduced functionality.
<code>audioRxStartMuted</code>	boolean	Whether audio from the endpoint is muted at the start of a call. Default: <code>false</code> .
<code>videoRxStartMuted</code>	boolean	Whether video from the endpoint is muted at the start of a call. Default: <code>false</code> .
<code>audioTxStartMuted</code>	boolean	Whether audio to the endpoint is muted at the start of a call. Default: <code>false</code> .
<code>videoTxStartMuted</code>	boolean	Whether video to the endpoint is muted at the start of a call. Default: <code>false</code> .
<code>autoReconnect</code>	boolean	Whether outgoing calls dropped for abnormal reasons are reconnected. Not effective for incoming calls. This setting will be ignored if <code>alwaysReconnect</code> is <code>true</code> . Default: <code>false</code> .
<code>recordingDevice</code>	boolean	Whether this call is treated as a recording device by muting received video and displaying a recording icon on other endpoints. Default: <code>false</code> .
<code>recordingDeviceIndicateOnly</code>	boolean	Whether this call is indicated as a recording device by displaying a recording icon on other endpoints. Received video is not muted. Default: <code>false</code> .
<code>deferConnect</code>	boolean	Applies only to calls dialed out from the TelePresence Server.  If <code>true</code> , the TelePresence Server defers connecting this call until at least one other call is connected to the conference. If <code>false</code> , the TelePresence Server connects the call as soon as the conference starts. Default: <code>false</code> .

**Table 32** callAttributes struct members (continued)

Parameter name	Type	Description
alwaysReconnect	boolean	If <code>true</code> , the TelePresence Server will reconnect this call in all disconnection circumstances. Applies only to calls dialed out from the TelePresence Server. Default: <code>false</code> .  <b>Caution:</b> This feature is intended for reconnecting integrated systems. Do not use it directly with user endpoints, as they will always be redialed even after deliberate disconnection.
iXEnabled	boolean	Defines whether the iX protocol is enabled on this call. This attribute is effective from the start of the call and cannot be changed during the call. Default: <code>true</code> .
displayLayoutSwitchingMode	string	Defines how the display of this multicamera system is switched into view on three-screen endpoints. Either <code>switchingRoomSwitched</code> or <code>switchingSegmentSwitched</code> . Defaults to <code>switchingSegmentSwitched</code> . See <a href="#">Enumerated Types, page 26</a> .  <b>Note:</b> This has no effect on multistream calls.
indicateMuting	boolean	Defines whether the participant is shown an icon representing that their audio has been muted on the TelePresence Server. <code>true</code> shows the icon when the participant has been muted. Default: <code>true</code> .
allowStarSixMuting	boolean	Defines whether participants can mute or unmute their audio by pressing *6. <code>true</code> allows the participant to use the *6 combination to mute/unmute.  Default: <code>true</code> .
multistreamMode	string	Select if multistream is used for the call. see <a href="#">Table 21 Multistream mode enumerated type, page 30</a> . Default is <code>multistreamOn</code> .
displayOnScreenMessages	boolean	Display on screen messages. Default: <code>true</code> .

## Conference URI Details Struct

The conference URI details struct defines a conference URI and its associated access levels and media resources.

**Table 33** Conference Connection Definitions

Parameter name	Type	Description
URI	string (80)	<b>Required.</b> String used by endpoints to connect to this conference. See <a href="#">Conference URI Identifiers, page 17</a> .

**Table 33 Conference Connection Definitions (continued)**

Parameter name	Type	Description
<code>callBandwidth</code>	integer ( $\geq$ <code>minCallBandwidth</code> and $\leq$ <code>maxCallBandwidth</code> )	<b>Required.</b> Connection bandwidth measured in bits per second. See <a href="#">flex.resource.query</a> , page 108.
<code>PIN</code>	string (40)	PIN for the conference at this URI and access level. Participants will only need to supply this PIN when calling in to the associated conference URI. A PIN is never requested when the TelePresence Server calls out to an endpoint. Default: empty.
<code>callAttributes</code>	struct	This <a href="#">Call Attributes Struct</a> , page 34 contains attributes applied to calls connecting to a conference using the aforementioned URI.  See <a href="#">Call Attributes Struct</a> , page 34 and <a href="#">How Call Attributes are Derived</a> , page 34.  Default: inherits the conference default call attributes, see <a href="#">flex.conference.create</a> , page 57 and <a href="#">flex.conference.modify</a> , page 67.
<code>participantMediaResources</code>	struct	The <a href="#">Participant Media Resources Struct</a> , page 33 contains parameters that define the participant media resource configuration.

If `participantMediaResources` settings are absent, settings from the conference default `participantMediaResources` apply.

### Participant Call Definition Struct

A single participant call can be defined as one of incoming or outgoing, but not both. As a result, there are two call definition structs: one for incoming calls and the other for outgoing calls.

**Table 34 Incoming participant call definition struct members**

Parameter name	Type	Description
<code>URI</code>	string (80)	<b>Required.</b> String used to connect to this conference. See <a href="#">Participant Conference URIs</a> , page 18.
<code>callBandwidth</code>	integer ( $\geq$ <code>minCallBandwidth</code> and $\leq$ <code>maxCallBandwidth</code> )	<b>Required.</b> Connection bandwidth measured in bits per second. See <a href="#">flex.resource.query</a> , page 108.
<code>disconnectOnIncoming</code>	boolean	Whether the existing call (if one exists) is disconnected when an incoming call comes through the <code>URI</code> . Default: <code>false</code> .

**Table 35 Outgoing participant call definition struct members**

Parameter name	Type	Description
<code>remoteAddress</code>	string (80)	<b>Required.</b> Address of the endpoint expressed in the form of an endpoint address or E164 number.



**Table 35** Outgoing participant call definition struct members (continued)

Parameter name	Type	Description
<code>protocol</code>	string	<b>Required.</b> Call control protocol for outgoing call only. See <a href="#">Enumerated Types, page 26</a> .
<code>callBandwidth</code>	integer ( $\geq$ <code>minCallBandwidth</code> and $\leq$ <code>maxCallBandwidth</code> )	<b>Required.</b> Connection bandwidth in bits per second. See <a href="#">flex.resource.query, page 108</a> .
<code>toOverride</code>	String	<b>Optional.</b> The SIP To-URI will be overridden with this URI, if supplied. Up to 80 characters. Used for outgoing SIP calls only.

## Participant PIN Definition

For a participant with any incoming calls, up to two PINs may be supplied for the specified URI(s).

**Table 36** Incoming participant PIN definition

Parameter name	type	Description
<code>PIN</code>	String	<b>Required.</b> PIN for this participant URI, up to 40 characters long.
<code>callAttributes</code>	callAttributes struct	Attributes applied to calls connecting to conference using the aforementioned participant URI (access level, etc.). Settings defined here override settings defined at conference default call attributes. Values of undefined members are considered unset. See <a href="#">Call Attributes Struct, page 34</a> .

## Enumerated Reservation Information Struct (experimental only—do not use)

**Table 37** Enumerated reservation information struct members

Parameter name	Type	Description
<code>reservationID</code>	string (50)	Reservation identifier, assigned by the DSC.
<code>reservationReference</code>	string (50)	Client reference string.
<code>conferenceID</code>	string (50)	Identifier of the conference the reservation is associated with.
<code>participantRole</code>	string	What role the reservation is for.
<code>numParticipants</code>	integer ( $\geq$ 1)	Number of participants the reservation is for.
<code>maxQuality</code>	integer ( $\geq$ 0)	Maximum quality (in total credits) of each participant.
<code>totalReservedCredits</code>	integer ( $\geq$ 0)	Total amount of credits reserved by the reservation.
<code>numParticipantsRemaining</code>	integer ( $\geq$ 0)	Number of participant still capable of using the reservation.
<code>numCreditsRemaining</code>	integer ( $\geq$ 0)	Number of credits still reserved by the reservation.

## Flexible API Command Reference

This section contains a reference to each of the commands available when the operation mode is set to `flexible`.

The commands are grouped alphabetically by the objects that they query or modify. The following information is provided for each command:

- Description of the command's effect
- Accepted parameters, and whether they are required
- Returned parameters, and whether they are conditionally returned

Click the command name to read a detailed description of the command.

- [callHome.configure](#), page 43
- [callHome.query](#), page 44
- [cdrlog.enumerate](#), page 44
- [cdrlog.query](#), page 45
- [device.feature.add](#), page 46
- [device.feature.remove](#), page 46
- [device.health.query](#), page 46
- [device.network.modify](#), page 47
- [device.network.query](#), page 48
- [device.query](#), page 50
- [device.restart](#), page 51
- [device.restartlog.query](#), page 52
- [device.time.modify](#), page 52
- [device.time.query](#), page 53
- [feedbackReceiver.configure](#), page 53
- [feedbackReceiver.query](#), page 54
- [feedbackReceiver.reconfigure](#), page 55
- [feedbackReceiver.remove](#), page 55
- [feedbackReceiver.status](#), page 55
- [flex.call.status](#), page 56
- [flex.conference.create](#), page 57
- [flex.conference.deletions.enumerate](#), page 64
- [flex.conference.destroy](#), page 64
- [flex.conference.enumerate](#), page 64
- [flex.conference.getMetadata](#), page 67
- [flex.conference.modify](#), page 67
- [flex.conference.query](#), page 72
- [flex.conference.sendUserMessage](#), page 78
- [flex.conference.sendWarning](#), page 78
- [flex.conference.status](#), page 78
- [flex.licenseMode.modify](#), page 79
- [flex.licenseMode.verify](#), page 80
- [flex.participant.advanced.enumerate](#), page 80
- [flex.participant.call.disconnect](#), page 83
- [flex.participant.clearImportant](#), page 83
- [flex.participant.create](#), page 84
- [flex.participant.deletions.enumerate](#), page 86

- [flex.participant.destroy](#), page 87
- [flex.participant.enumerate](#), page 87
- [flex.participant.media.enumerate](#), page 89
- [flex.participant.modify](#), page 91
- [flex.participant.query](#), page 91
- [flex.participant.requestDiagnostics](#), page 93
- [flex.participant.requestPreview](#), page 99
- [flex.participant.sendDTMF](#), page 101
- [flex.participant.sendUserMessage](#), page 102
- [flex.participant.setImportant](#), page 102
- [flex.participant.setMute](#), page 102
- [flex.participant.status](#), page 103
- [flex.resource.query](#), page 108
- [flex.resource.status](#), page 110
- [logs.syslog.modify](#), page 111
- [logs.syslog.query](#), page 111
- [services.modify](#), page 111
- [services.query](#), page 112
- [sip.modify](#), page 113
- [sip.query](#), page 114
- [system.info](#), page 114
- [user.create](#), page 116
- [user.destroy](#), page 116
- [user.enumerate](#), page 117
- [user.modify](#), page 117

## callHome.configure

Configures the TelePresence Server to automatically report diagnostic data to Cisco's Call Home service. This feature is disabled by default, but we strongly recommend that you enable it to ensure the best possible support for your device.

**Note:** The TelePresence Server currently only supports anonymous reporting.

**Table 38** `callHome.configure` inputs

Parameter name	Type	Description
<code>mode</code>	string	<p>Set the Call Home mode. One of <code>disabled</code> or <code>anonymous</code>.</p> <p>Can only be set to <code>anonymous</code> if the encryption feature key is present. Defaults to <code>disabled</code> if it has never been configured.</p> <p>Omit the parameter to leave the current setting unchanged.</p>

**Table 38** `callHome.configure` inputs (continued)

Parameter name	Type	Description
<code>automatic</code>	boolean	Controls automatic Call Home.  <code>true</code> enables automatic Call Home. <code>false</code> disables automatic Call Home. Only has effect when mode is <code>anonymous</code> .  Omit the parameter to leave the current setting unchanged.

## `callHome.query`

Queries the TelePresence Server to retrieve its Call Home configuration. This feature reports diagnostic data to Cisco's Call Home service.

**Note:** The TelePresence Server currently only supports anonymous reporting.

**Table 39** `callHome.query` returned data

Parameter name	Type	Description
<code>mode</code>	string	Call Home mode. One of <code>disabled</code> or <code>anonymous</code> . Defaults to <code>disabled</code> if it has never been configured.
<code>automatic</code>	boolean	<code>true</code> if automatic Call Home is enabled. <code>false</code> if automatic Call Home is disabled. Only has effect if mode is <code>anonymous</code> .  Defaults to <code>false</code> if it has never been configured.

## `cdrlog.enumerate`

This call allows the calling application to download CDR log data without having to return the entire CDR log. The call returns a subset of the CDR log based on the optional `filter`, `index` and `numEvents` parameters.

TelePresence Server holds up to 2000 records in memory. It does not permanently retain these, so we recommend that your application either makes regular enumerate calls or triggers enumerate calls upon receiving the `cdrAdded` feedback event.

**Table 40** `cdrlog.enumerate` optional or conditional inputs

Parameter name	Type	Description
<code>index</code>	integer	Index from which to get events. The device returns the <code>nextIndex</code> so the application can use it to retrieve the next enumeration of CDR data.  If <code>index</code> is omitted, negative, or greater (by 2 or more) than the highest index, the device will enumerate events from the beginning of the CDR log.
<code>numEvents</code>	integer	Maximum number of events to be returned per enumeration. If omitted (or not between 1–20 inclusive), a maximum of 20 events will be returned per enumeration.  Fewer events are returned if they are too large to fit into a single response. Clients should look at the <code>eventsRemaining</code> parameter in the response and re-enumerate, starting from <code>nextIndex</code> , if necessary.

**Table 40** `cdrlog.enumerate` optional or conditional inputs (continued)

Parameter name	Type	Description
<code>filter</code>	array	An array of strings, which contain the names of event types by which to filter the response. Omit <code>filter</code> to return all event types or include a subset of the following: <code>conferenceStarted</code> , <code>conferenceFinished</code> , <code>conferenceActive</code> , <code>conferenceInactive</code> , <code>participantConnected</code> , <code>participantJoined</code> , <code>participantMediaSummary</code> , <code>participantLeft</code> , <code>participantDisconnected</code> .

**Table 41** `cdrlog.enumerate` returned data

Parameter name	Type	Description
<code>startIndex</code>	integer	Either the index provided, or if that is lower than the index of the first record the device has, it will be the first record it does know about. In this case, comparing the <code>startIndex</code> with the index provided gives the number of dropped records.
<code>nextIndex</code>	integer	Revision number of the data being provided, reusable in a subsequent call to the API.
<code>eventsRemaining</code>	boolean	If <code>true</code> , there is more data in the requested enumeration than has been returned in this response.
<code>currentTime</code>	dateTime.iso8601	The system's current time (UTC).
<code>events</code>	array of structs	Each member of the array is a struct that represents a recorded event. The structures all have some common fields ( <code>time</code> , <code>type</code> , <code>index</code> ) and may have other fields that are specific to the event type.

## Events array

The following parameters are common to all CDR log events, but each struct will also contain parameters specific to the event type. See [Cisco TelePresence Conferencing Call Detail Records File Format Reference Guide](#) for details of all the TelePresence Server's event types.

If there are no events to enumerate, the `events` array is returned empty.

**Table 42** Common CDR log event parameters

Parameter name	Type	Description
<code>time</code>	dateTime.iso8601	Date and time when the event was logged; for example, <code>20110119T13:52:42</code> .
<code>type</code>	string	Name of the event type.
<code>index</code>	integer	Index of the CDR log message.

**Note:** The [Cisco TelePresence Conferencing Call Detail Records File Format Reference Guide](#) describes the CDR log in its XML form, as downloaded in `cdr_log.xml` via the web interface. When the same events are enumerated with this call, the event type names use camelCase for multiple words rather than using underscores. For example, `conference_started` in `cdr_log.xml` is the same event type as `conferenceStarted` in this array.

## cdrlog.query

Returns information about the CDR log. This command takes no input parameters.

**Table 43** `cdrlog.query` returned data

Parameter name	Type	Description
<code>firstIndex</code>	integer	Index of the oldest stored event.
<code>numEvents</code>	integer	Total number of events stored.

## device.feature.add

Adds a license or feature to the TelePresence Server. You need to obtain a key from Cisco or one of its resellers prior to running this command.

**Table 44** `device.feature.add` inputs

Parameter name	Type	Description
<code>key</code>	string	<b>Required.</b> Use this unique code when you wish to add conferencing capacity or an optional feature to your TelePresence Server.

## device.feature.remove

Removes a license or feature from the TelePresence Server. Use `device.query` to read the keys from a TelePresence Server.

**Table 45** `device.feature.remove` inputs

Parameter name	Type	Description
<code>key</code>	string	<b>Required.</b> The unique code associated with the optional feature or license that you wish to remove from the TelePresence Server.

## device.health.query

Returns the current status of the device, such as health monitors and CPU load. This command takes no input parameters.

**Table 46** `device.health.query` returned data

Parameter name	Type	Description
<code>cpuLoad</code>	integer	CPU load expressed as a percentage of the maximum.
<code>fanStatus</code>	string	<code>ok</code> or <code>outOfSpec</code> . This parameter is returned only on appliances, eg. Media 310 or TelePresence Server 7010, which have their own fans. This parameter is not returned for TelePresence Server blades.
<code>fanStatusWorst</code>	string	Worst fan status recorded on this device since it restarted. One of <code>ok</code> or <code>outOfSpec</code> .  This parameter is returned only on appliances, eg. Media 310 or TelePresence Server 7010, which have their own fans. This parameter is not returned for TelePresence Server blades.

**Table 46** `device.health.query` returned data (continued)

Parameter name	Type	Description
<code>temperatureStatus</code>	string	One of <code>ok</code> (the temperature is currently within the normal operating range), <code>outOfSpec</code> (the temperature is currently outside the normal operating range), or <code>critical</code> (the temperature is too high and the device will shutdown if this condition persists).
<code>temperatureStatusWorst</code>	string	Worst temperature status recorded on this device since it booted. One of <code>ok</code> , <code>outOfSpec</code> , or <code>critical</code> .
<code>rtcBatteryStatus</code>	string	Current status of the RTC battery (Real Time Clock). One of <code>ok</code> , <code>outOfSpec</code> , or <code>critical</code> .
<code>rtcBatteryStatusWorst</code>	string	Worst status of the RTC battery (Real Time Clock) recorded on this device since it booted. One of <code>ok</code> , <code>outOfSpec</code> , or <code>critical</code> .
<code>voltagesStatus</code>	string	One of <code>ok</code> (the voltage is currently within the normal range), <code>outOfSpec</code> (the voltage is currently outside the normal range), or <code>critical</code> .
<code>voltagesStatusWorst</code>	string	Worst voltage status recorded on this device since it booted. One of <code>ok</code> , <code>outOfSpec</code> , or <code>critical</code> .
<code>operationalStatus</code>	string	One of <code>active</code> (the device is active), <code>shuttingDown</code> (the device is shutting down), <code>shutDown</code> (the device has shut down), or <code>unknown</code> .

## device.network.modify

This call changes the TelePresence Server port configuration settings.

**Note:** If this API command is used to change the IP address which is being used to receive the API command and send the response, then the XML-RPC response may be wholly or partially lost, which may appear as a fail condition in the API client. It is however safe to change IPv4 settings while using the API over IPv6 or vice versa.

**Table 47** `device.network.modify` inputs

Parameter name	Type	Description
<code>portA</code>	struct	<b>Optional.</b> The struct contains configuration and status information for Ethernet port A on the device. See <a href="#">Table 48 Port struct members</a> , page 48.
<code>dns</code>	array of structs	<b>Optional.</b> Each member of the array is a struct representing a set of DNS parameters for the device. See <a href="#">Table 49 DNS struct members</a> , page 48.

Note that some fields must be configured together:

- `dhcpv4` is ignored if `ipv4Enabled` is not specified
- `ipv6Conf` is ignored if `ipv6Enabled` is not specified
- If `ipv4Enabled` is `True`, `dhcpv4` is required
- If `ipv4Enabled` is `True` and `dhcpv4` is `False` then `ipv4Address` and `ipv4SubnetMask` are required
- If `ipv6Enabled` is `True`, `ipv6Conf` is required
- If `ipv6Enabled` is `True` and `ipv6Conf` is `manual` then `ipv6Address` and `ipv6PrefixLength` are required

**Table 48 Port struct members.**

Parameter name	Type	Description
<code>ipv4Enabled</code>	boolean	True if IPv4 interface is enabled.
<code>dhcpv4</code>	boolean	True if DHCP is used to set the IPv4 address details. If 'true', then static IPv4 details in this API can be set but will not take effect.
<code>ipv4Address</code>	string	IPv4 address in the dotted quad format.
<code>ipv4SubnetMask</code>	string	IPv4 subnet mask in the dotted quad format.
<code>defaultIpv4Gateway</code>	string	IPv4 address in the dotted quad format.
<code>ipv6Enabled</code>	boolean	Whether IPv6 interface is enabled. Always returned unless there are no IP interfaces enabled on the port (neither IPv4 nor IPv6 is enabled).
<code>ipv6Address</code>	string	IPv6 address in CIDR format. Not returned if not configured.
<code>ipv6Conf</code>	string	Indicates how the IPv6 address is assigned. One of <code>automatic</code> (IPv6 address is configured by SLAAC/DHCPv6) or <code>manual</code> (IPv6 address is configured manually).  Not returned if not configured.
<code>ipv6PrefixLength</code>	integer	Length of the IPv6 address prefix.
<code>defaultIpv6Gateway</code>	string	Address of the IPv6 default gateway in CIDR format.

**Table 49 DNS struct members**

Parameter name	Type	Description
<code>dnsConfiguration</code>	string	How the TelePresence Server is acquires its name server address. One of <code>portAIPv4</code> , <code>portAIPv6</code> , or <code>manual</code> . If <code>manual</code> , then at least one name server must be provided.
<code>hostName</code>	string	Host name of the device.
<code>nameServer</code>	string	IP address of the name server, in dotted quad format (IPv4) or CIDR format (IPv6).
<code>nameServerSecondary</code>	string	IP address of the secondary name server, in dotted quad format (IPv4) or CIDR format (IPv6).
<code>domainName</code>	string	Domain name of the device (DNS suffix).

## device.network.query

Queries the device for its network information. The call takes no parameters and returns the following data structures. Some of the data listed below will be omitted if the interface is not enabled or configured. The query returns empty strings or dashes for addresses that are not configured.

**Note:** Packet counts and other statistics are measured with 32-bit signed integers, and may therefore wrap.



**Table 50** `device.network.query` returned data

Parameter name	Type	Description
<code>portA</code>	struct	The struct contains configuration and status information for Ethernet port A on the device. See <a href="#">Table 51 Port struct members, page 49</a> .
<code>dns</code>	array of structs	Each member of the array is a struct representing a set of DNS parameters for the queried device. See <a href="#">Table 52 DNS struct members, page 50</a> .

**Table 51** Port struct members

Parameter name	Type	Description
<code>enabled</code>	boolean	Whether the port is enabled.
<code>ipv4Enabled</code>	boolean	Whether IPv4 interface is enabled. Always returned unless there are no IP interfaces enabled on the port (neither IPv4 nor IPv6 is enabled).
<code>ipv6Enabled</code>	boolean	Whether IPv6 interface is enabled. Always returned unless there are no IP interfaces enabled on the port (neither IPv4 nor IPv6 is enabled).
<code>linkStatus</code>	boolean	Whether the Ethernet connection to this port is active.
<code>speed</code>	integer	Speed of the connection on this Ethernet port. One of 10, 100 or 1000, in Mbps.
<code>fullDuplex</code>	boolean	Whether the port can support a full-duplex connection.
<code>macAddress</code>	string	MAC address of this port. A 12-character string of hex digits with no separators.
<code>packetsSent</code>	integer	Number of packets sent from this Ethernet port.
<code>packetsReceived</code>	integer	Number of packets received on this Ethernet port.
<code>multicastPacketsSent</code>	integer	Number of multicast packets sent from this Ethernet port.
<code>multicastPacketsReceived</code>	integer	Number of multicast packets received on this Ethernet port.
<code>bytesSent</code>	integer	Number of bytes sent by the device.
<code>bytesReceived</code>	integer	Number of bytes received by the device.
<code>queueDrops</code>	integer	Number of packets dropped from the queue on this network port.
<code>collisions</code>	integer	Count of the network collisions recorded by the device.
<code>transmitErrors</code>	integer	Count of transmission errors on this Ethernet port.
<code>receiveErrors</code>	integer	Count of receive errors on this port.
<code>bytesSent64</code>	string	64-bit versions of the <code>bytesSent</code> statistic expressed as a string rather than an integer.
<code>bytesReceived64</code>	string	64-bit versions of the <code>bytesReceived</code> statistic expressed as a string rather than an integer.

**Table 51 Port struct members (continued)**

Parameter name	Type	Description
<code>dhcpv4</code>	boolean	Whether the ipv4 address is allocated by DHCP. Not returned if not configured.
<code>ipv4Address</code>	string	IPv4 address in the dotted quad format. Not returned if not configured.
<code>ipv4SubnetMask</code>	string	IPv4 subnet mask in the dotted quad format. Not returned if not configured.
<code>defaultIpv4Gateway</code>	string	IPv4 address in the dotted quad format. Not returned if not configured.
<code>ipv6Address</code>	string	IPv6 address in CIDRformat. Not returned if not configured.
<code>ipv6Conf</code>	string	Indicates how the IPv6 address is assigned. One of <code>automatic</code> (IPv6 address is configured by SLAAC/DHCPv6) or <code>manual</code> (IPv6 address is configured manually).  Not returned if not configured.
<code>ipv6PrefixLength</code>	integer	Length of the IPv6 address prefix. Not returned if not configured.
<code>defaultIpv6Gateway</code>	string	Address of the IPv6 default gateway in CIDR format. Not returned if not configured.
<code>linkLocalIpv6Address</code>	string	Link local IPv6 address in CIDR format. Not returned if not configured.
<code>linkLocalIpv6PrefixLength</code>	integer	Length of the link local IPv6 address prefix. Not returned if not configured.

**Table 52 DNS struct members**

Parameter name	Type	Description
<code>dnsConfiguration</code>	string	How the TelePresence Server acquires its name server address. One of <code>portAIPv4</code> , <code>portAIPv6</code> , or <code>manual</code> .
<code>hostName</code>	string	Host name of the queried device.
<code>nameServer</code>	string	IP address of the name server, in dotted quad format (IPv4) or CIDR format (IPv6).
<code>nameServerSecondary</code>	string	IP address of the secondary name server, in dotted quad format (IPv4) or CIDR format (IPv6).
<code>domainName</code>	string	Domain name of the queried device (DNS suffix).

## device.query

Returns high level status information about the device. This command takes no input parameters.

**Table 53 device.query returned data**

Parameter name	Type	Description
<code>currentTime</code>	<code>dateTime.iso8601</code>	The system's current date and time.
<code>restartTime</code>	<code>dateTime.iso8601</code>	The system's date and time when it started.

**Table 53** `device.query` returned data (continued)

Parameter name	Type	Description
<code>uptime</code>	integer	The difference, in seconds, between the system's current time and the system's restart time.
<code>serial</code>	string	Serial number of this device.
<code>apiVersion</code>	string	Version number of the API implemented by this TelePresence Server.
<code>activatedFeatures</code>	array of structs	Each member of the array is a struct, representing an active feature. See <a href="#">Table 54 Active feature struct members, page 51</a> .
<code>activatedLicenses</code>	array of structs	Each member of the array is a struct, representing an active license. See <a href="#">Table 55 Active license struct members, page 51</a> .
<code>shutdownStatus</code>	string	Displays one of the following: <code>notShutdown</code> , <code>shutdownInProgress</code> , <code>shutdown</code> , or <code>error</code> .
<code>mediaResourceRestarts</code>	integer	The count of unexpected restarts that have occurred on the device's media resources (signal processor chips).

**Table 54** Active feature struct members

Parameter name	Type	Description
<code>feature</code>	string	The name of the feature, eg. <code>Encryption</code> .
<code>key</code>	string	The unique code associated with the feature.
<code>expiry</code>	<code>dateTime.iso8601</code>	The time at which this temporary key will expire. <code>expiry</code> is not present for permanent keys.

**Table 55** Active license struct members

Parameter name	Type	Description
<code>license</code>	string	The name of the license.
<code>ports</code>	integer	The number of screen licenses provided by this license.
<code>key</code>	string	The unique code associated with the license.
<code>expiry</code>	<code>dateTime.iso8601</code>	The time at which this temporary key will expire. <code>expiry</code> is not present for permanent keys.

## `device.restart`

Restarts the device, or shuts it down without a restart. This command does not return any parameters.

**Table 56** `device.restart` input parameters

Parameter name	Type	Description
<code>shutdownOnly</code>	boolean	(Optional) Set to <code>true</code> to shut down without restarting. Default: <code>false</code> .

`device.restartlog.query`

Returns the restart log - also known as the system log on the web interface. This command takes no input parameters.

**Table 57** `device.restartlog.query` returned data

Parameter name	Type	Description
<code>log</code>	array of structs	Each member of the array is a struct containing a restart <code>reason</code> . See <a href="#">Table 58 Log struct members, page 52</a> .  This information source is called "system log" in the web interface.

**Table 58** Log struct members

Parameter name	Type	Description
<code>time</code>	<code>dateTime.iso8601</code>	Date and time of the restart.
<code>reason</code>	string	Reason for the device restart. See <a href="#">Table 59 Restart reason enumerated type, page 52</a> .

**Table 59** Restart `reason` enumerated type

<code>reason</code> value	Description
User requested shutdown	The device restarted normally after a user initiated a shutdown.
User requested reboot from web interface	The device restarted itself because a user initiated a reboot via the web interface.
User requested upgrade	The device restarted itself because a user initiated an upgrade.
User requested reboot from console	The device restarted itself because a user initiated a reboot via the console.
User requested reboot from API	The device restarted itself because a user initiated a reboot via the API.
User requested reboot from FTP	The device restarted itself because a user initiated a reboot via FTP.
User requested shutdown from supervisor	The device restarted normally after a user initiated a shutdown from the supervisor.
User requested reboot from supervisor	The device restarted itself because a user initiated a reboot via the supervisor.
User reset configuration	The device restarted itself because a user reset the configuration.
Cold boot	The device restarted itself because a user initiated a cold boot.
unknown	The software is unaware why the device restarted.

`device.time.modify`

Modifies the time settings of the device.

**Table 60** `device.time.modify` inputs

Parameter name	Type	Description
<code>currentTime</code>	date Time ISO8601	The current time (UTC). Cannot be modified if NTP is enabled.
<code>ntpEnabled</code>	boolean	True to enable NTP. In which case <code>ntpHost</code> must be supplied.
<code>utcOffsetHours</code>	Integer	Range -12 to +14 inclusive. Together with <code>utcOffsetMinutes</code> specifies offset to UTC.
<code>utcOffsetMinutes</code>	Integer	Range 0 to 59 inclusive. Together with <code>utcOffsetHours</code> specifies offset to UTC.
<code>ntpHost</code>	string	DNS or IP address of an NTP server.

`device.time.query`

Returns the current time settings of the device.

**Table 61** `device.time.query` returned data

Parameter name	Type	Description
<code>currentTime</code>	date Time ISO8601	The current time (UTC).
<code>ntpEnabled</code>	boolean	True if NTP is enabled.
<code>utcOffsetHours</code>	Integer	Range -12 to +14 inclusive. Together with <code>utcOffsetMinutes</code> specifies offset to UTC.
<code>utcOffsetMinutes</code>	Integer	Range 0 to 59 inclusive. Together with <code>utcOffsetHours</code> specifies offset to UTC.
<code>ntpHost</code>	string	DNS or IP address of an NTP server.
<code>ntpStatus</code>	string	The NTP client's current status; one of <code>disabled</code> , <code>synchronizing</code> , <code>synchronized</code> or <code>error</code> .

`feedbackReceiver.configure`

Configures the device to send feedback about the specified `subscribedEvents` to the specified `receiverURI`.

**Table 62** `feedbackReceiver.configure` inputs

Parameter name	Type	Description
<code>receiverURI</code>	string (255)	<b>Required.</b> Fully-qualified <code>http</code> or <code>https</code> URI (for example, <code>http://tms1:8080/RPC2</code> ) to which feedback events are sent. If no port number is specified, the device uses the protocol defaults (80 and 443 respectively).

**Table 62** `feedbackReceiver.configure` inputs (continued)

Parameter name	Type	Description
<code>receiverIndex</code>	integer ( $< 0$ , or 1–20 inclusive)	Index of the feedback receiver indicating the slot that this receiver should use. A negative value indicates that the feedback receiver should use any available slot (preferred). Default: 1.  <b>Note:</b> The default <code>receiverIndex</code> is 1, and will always overwrite a feedback receiver in the first index position. You should query the device first, or use a negative value, if you want to be certain not to overwrite an existing feedback receiver.
<code>sourceIdentifier</code>	string (255 ASCII characters only)	Identifier string for the receiver. The originating device uses this parameter to identify itself to the listening receiver (or receivers). If the parameter is not explicitly set, the device identifies itself with the MAC address of its Ethernet port A interface. Default: empty.
<code>subscribedEvents</code>	array	An array of strings, each of which is the name of a notification event. The array defines the events to which the receiver subscribes. See <a href="#">Feedback Events, page 24</a> . If this array is absent, the receiver subscribes to all notifications by default. Default: all events.

**Table 63** `feedbackReceiver.configure` returned data

Parameter name	Type	Description
<code>receiverIndex</code>	integer	Position of this feedback receiver in the device's table of feedback receivers.

## `feedbackReceiver.query`

Requests a list of all the feedback receivers that have previously been configured for the device. It does not accept parameters other than the authentication strings. If there are no feedback receivers to enumerate, `feedbackReceiver.query` returns an empty `receivers` array.

**Table 64** `feedbackReceiver.query` returned data

Parameter name	Type	Description
<code>receivers</code>	array	Array of feedback receivers, with members corresponding to the entries in the receivers table on the web interface of the device.

**Table 65** Feedback receiver struct members

Parameter name	Type	Description
<code>index</code>	integer (1–20)	Position of this feedback receiver in the table of feedback receivers. The index number is also the feedback receiver ID.
<code>sourceIdentifier</code>	string (255 ASCII characters only)	Source identifier string, which can be empty. The originating device uses this parameter to identify itself to the listening receiver (or receivers). If the parameter is not explicitly set, the device identifies itself with the MAC address of its Ethernet port A interface.
<code>receiverURI</code>	string (255)	Fully-qualified <code>http</code> or <code>https</code> URI (for example, <code>http://tms1:8080/RPC2</code> ) to which feedback events are sent.

## feedbackReceiver.reconfigure

Overwrites the configuration of an existing feedback receiver with any parameters that you supply. The TelePresence Server keeps the current configuration for any parameters that you do not specify.

**Table 66** `feedbackReceiver.reconfigure` inputs

Parameter name	Type	Description
<code>receiverIndex</code>	integer (1-20)	<b>Required.</b> Index of the feedback receiver to be reconfigured. The call returns a fault if there is no feedback receiver at the specified <code>receiverIndex</code> .
<code>receiverURI</code>	string (255)	Fully-qualified <code>http</code> or <code>https</code> URI (for example, <code>http://tms1:8080/RPC2</code> ) to which feedback events are sent. If omitted, the device uses the originally configured <code>receiverURI</code> .
<code>sourceIdentifier</code>	string (255) ASCII characters only	Identifier string for the receiver. The originating device uses this parameter to identify itself to the listening receiver (or receivers). If omitted, the device uses the originally configured <code>sourceIdentifier</code> .
<code>subscribedEvents</code>	array	Array of strings identifying the events to which the receiver subscribes. See <a href="#">Feedback Events, page 24</a> . If omitted, the event notifications set in the original configuration request remain unchanged.

## feedbackReceiver.remove

Removes the specified feedback receiver. This command returns no data.

**Table 67** `feedbackReceiver.remove` inputs

Parameter name	Type	Description
<code>receiverIndex</code>	integer (1-20)	<b>Required.</b> Index of the feedback receiver to be removed.

## feedbackReceiver.status

Asks the device for a list of all the events to which a feedback receiver subscribes.

**Table 68** `feedbackReceiver.status` inputs

Parameter name	Type	Description
<code>receiverIndex</code>	integer (1-20)	<b>Required.</b> Index of the feedback receiver.

**Table 69** `feedbackReceiver.status` returned data

Parameter name	Type	Description
<code>receiverIndex</code>	integer (1-20)	Index of the feedback receiver entry, which also serves as the feedback receiver ID.
<code>sourceIdentifier</code>	string (255) ASCII characters only	Identifier string for the receiver. The originating device uses this parameter to identify itself to the listening receiver (or receivers). If the parameter is not explicitly set, the device identifies itself with the MAC address of its Ethernet port A interface.

**Table 69** feedbackReceiver.status returned data (continued)

Parameter name	Type	Description
receiverURI	string (255)	Fully-qualified http or https URI (for example, http://tms1:8080/RPC2) to which feedback events are sent.
subscribedEvents	array	Array of strings identifying the event names that are enabled for this feedback receiver. See <a href="#">Feedback Events, page 24</a> .

## flex.call.status

Returns the status of the specified call.

**Table 70** flex.call.status inputs

Parameter name	Type	Description
callID	string (50)	<b>Required.</b> Call identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .

**Table 71** flex.call.status returned data

Parameter name	Type	Description
callID	string (50)	Call identifier. See <a href="#">Identifiers and Client References, page 16</a> .
conferenceID	string (50)	Identifier of the conference to which the call is connected or is in the process of connecting. See <a href="#">Identifiers and Client References, page 16</a> .
conferenceState	string	State of call connection to the conference. See <a href="#">Table 16 Call conference state enumerated type, page 28</a> .
callState	string	State of the call. See <a href="#">Table 17 Call state enumerated type, page 28</a> .
incoming	boolean	Direction of the call: <code>true</code> indicates incoming; <code>false</code> indicates outgoing.
protocol	string	Call control protocol. See <a href="#">Table 12 Call protocol enumerated type, page 27</a> .
address	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.
participantID	string (50)	Participant identifier. See <a href="#">Identifiers and Client References, page 16</a> . Returned if string length > 0.
duration	integer (>= 0)	Duration of the call, in seconds. Only returned if a call has been established.
rxBandwidth	integer (>= 0)	Receive bandwidth. Only returned if a call has been established.
txBandwidth	integer (>= 0)	Transmit bandwidth. Only returned if a call has been established.
remoteName	string (80)	Endpoint name supplied by the far end. Returned if string length > 0.



## flex.conference.create

Creates a conference with the supplied parameters and returns the unique identifier of the new conference.

The following parameters must be present for this command to succeed:

- `participantMediaResources.mediaTokensMainVideo.total`
- `participantMediaResources.mediaTokensExtendedVideo.total`
- `participantMediaResources.mediaTokensAudio.total`
- `participantMediaResources.numMediaCredits`

### Custom Lobby Screen Disclaimer

The custom lobby screen background feature in TelePresence Server is provided as a preview feature and is not intended for use in production environments. Use of the custom lobby screen background feature is subject to the software license and limited warranty

[http://www.cisco.com/c/en/us/td/docs/general/warranty/English/EU1KEN\\_.html](http://www.cisco.com/c/en/us/td/docs/general/warranty/English/EU1KEN_.html) for TelePresence Server. Cisco reserves the right to disable the custom lobby screen background feature at any time without notice. Cisco Technical Support will provide limited assistance to customers who wish to use the custom lobby screen background feature.

**Table 72** `flex.conference.create` inputs

Parameter name	Type	Description
<code>participantMediaResources</code>	struct	Required. This <a href="#">Participant Media Resources Struct, page 33</a> defines the conference default participant media resource configuration.  These defaults can be overridden by the media resource configuration defined for the conference URI or for the participant.
<code>conferenceReference</code>	string (50)	Conference client reference string. See <a href="#">Identifiers and Client References, page 16</a> . Default: empty.
<code>conferenceName</code>	string (80)	Human readable label for the conference. Default: empty string.
<code>conferenceDescription</code>	string (500)	Human readable description of the conference. If set, the TelePresence Server uses this value for ActiveControl's conference description. Default: empty string.

Table 72 flex.conference.create inputs (continued)

Parameter name	Type	Description
<code>URIs</code>	array of structs	<p>Each member of the array is a <a href="#">Table 33 Conference Connection Definitions, page 39</a> defining a unique conference URI and its associated access levels and media tokens.</p> <p>You may supply a maximum of two conference URI structs. If you want two levels of access to the conference, for example for chairpersons and guests, you must create two structs in this array. Each struct in the array requires a unique URI.</p> <p>Default: empty array.</p> <p><b>Note:</b> The URI does not need to be unique, if certain conditions are met. See <a href="#">URIs Do Not Need to be Unique, page 18</a></p>
<code>conferenceMediaTokens</code>	integer (>= 0)	<p>Maximum number of media tokens that can be used for the conference.</p> <p>Do not supply <code>conferenceMediaTokens</code> if you supply <code>conferenceMediaTokensUnlimited</code>. This will result in a fault.</p> <p>Default: Absent. See <code>conferenceMediaTokensUnlimited</code>.</p>
<code>conferenceMediaTokensUnlimited</code>	boolean	<p>Whether no limit is defined for the number of media tokens that can be used for the conference.</p> <p>Do not supply <code>conferenceMediaTokensUnlimited</code> if you supply <code>conferenceMediaTokens</code>. This will result in a fault. See "<a href="#">Unlimited</a>" <a href="#">Integers, page 20</a>.</p> <p>Default: <code>true</code>.</p>
<code>conferenceMediaCredits</code>	integer (>= 0)	<p>Maximum number of media credits that can be used for the conference.</p> <p>Do not supply <code>conferenceMediaCredits</code> if you supply <code>conferenceMediaCreditsUnlimited</code>. This will result in a fault.</p> <p>Default: Absent. See <code>conferenceMediaCreditsUnlimited</code>.</p>
<code>conferenceMediaCreditsUnlimited</code>	boolean	<p>Whether no limit is defined for the number of media credits that can be used for the conference.</p> <p>Do not supply <code>conferenceMediaCreditsUnlimited</code> if you supply <code>conferenceMediaCredits</code>. This will result in a fault. See "<a href="#">Unlimited</a>" <a href="#">Integers, page 20</a>.</p> <p>Default: <code>true</code>.</p>
<code>waitForChair</code>	boolean	<p>Whether callers must wait for a chair to join the conference. Default: <code>true</code>.</p>

Table 72 flex.conference.create inputs (continued)

Parameter name	Type	Description
<code>disconnectOnChairExit</code>	boolean	Whether callers are disconnected when the last chair leaves the conference. Default: <code>false</code> .
<code>terminateWithLastCall</code>	boolean	Whether the conference is destroyed with the last call. Default: <code>false</code> .
<code>locked</code>	boolean	Whether the conference is locked, causing incoming calls to be rejected. Default: <code>false</code> .
<code>startTime</code>	integer (>= 0)	Number of seconds to wait before starting the conference. Default: 0.
<code>duration</code>	integer (>= 0)	Conference duration (in seconds) measured from the start time. If the conference duration is due to end in less than 120 seconds, participants are notified that it is about to end, as described in <a href="#">Conference send warning</a> .  Do not supply <code>duration</code> if you supply <code>durationUnlimited</code> . This will result in a fault.  Default: Absent. See <code>durationUnlimited</code> .
<code>durationUnlimited</code>	boolean	Whether an unlimited duration is assigned for the conference. If this field is present and the value is <code>true</code> , the value for the conference duration is ignored.  Do not supply <code>durationUnlimited</code> if you supply <code>duration</code> . This will result in a fault. See <a href="#">"Unlimited" Integers, page 20</a> .  Default: <code>true</code> .
<code>billingCode</code>	string (80)	Billing code string. Default: empty.
<code>callAttributes</code>	struct	Conference default call attributes. See <a href="#">How Call Attributes are Derived, page 34</a> . Default: see <a href="#">Table 32 callAttributes struct members, page 35</a> for defaults of struct members.
<code>maxParticipants</code>	integer (>= 0)	Maximum number of participants that can connect to this conference.  Do not supply <code>maxParticipants</code> if you supply <code>maxParticipantsUnlimited</code> . This will result in a fault.  Default: Absent. See <code>maxParticipantsUnlimited</code> .

Table 72 flex.conference.create inputs (continued)

Parameter name	Type	Description
<code>maxParticipantsUnlimited</code>	boolean	Whether an unlimited number of participants are allowed to connect to this conference.  Do not supply <code>maxParticipantsUnlimited</code> if you supply <code>maxParticipants</code> . This will result in a fault. See <a href="#">"Unlimited" Integers, page 20</a> .  Default: <code>true</code> .
<code>voiceSwitchingSensitivity</code>	integer (0-100)	Voice switching sensitivity. Default: 60.
<code>welcomeScreen</code>	boolean	Whether a welcome screen message is displayed for 5 seconds when a call joins the conference. See <code>welcomeScreenMessage</code> for contents of the message. Default: <code>true</code> .
<code>welcomeScreenMessage</code>	string (500)	Welcome screen message for this conference. If this string is empty, the conference name is displayed as the welcome screen message.  If <code>conferenceDescription</code> is not set, the TelePresence Server uses this value for ActiveControl's conference description. Default: empty string.
<code>useCustomPINEntryMessage</code>	boolean	Whether a custom message is displayed in the PIN entry screen. Default: <code>false</code> .
<code>customPINEntryMessage</code>	string (200)	Custom message for PIN entry. Only used if <code>useCustomPINEntryMessage</code> is <code>true</code> . Default: empty.
<code>useCustomOptionalPINEntryMessage</code>	boolean	Display custom message at shared PIN entry screen when conference can be entered without a PIN. Default: <code>false</code> .
<code>customOptionalPINEntryMessage</code>	string (200)	Custom message for shared PIN entry screen when conference can be entered without a PIN, if <code>useCustomOptionalPINEntryMessage</code> is <code>true</code> . Default: empty.
<code>useCustomPINIncorrectMessage</code>	boolean	Whether a custom warning message is displayed in the PIN entry screen after an incorrect PIN has been entered. Default: <code>false</code> .
<code>customPINIncorrectMessage</code>	string (100)	Custom warning message for incorrect PIN entry. Only used if <code>useCustomPINIncorrectMessage</code> is <code>true</code> . Default: empty.
<code>useCustomWaitingForChairMessage</code>	boolean	Whether a custom message is displayed when waiting for a chair to join the conference. Default: <code>false</code> .
<code>customPINEntryFailedMessage</code>	string (200)	Optional. Custom message displayed on the exit lobby if a participant is disconnected for failing to enter the PIN three times.

Table 72 flex.conference.create inputs (continued)

Parameter name	Type	Description
<code>useCustomPINEntryFailedMessage</code>	boolean	Optional. Whether the custom message is displayed for the PIN failure exit lobby.
<code>customDisconnectPlatitudeMessage</code>	string (200)	Optional. Custom "thank you" message displayed on all exit lobbies.
<code>useCustomDisconnectPlatitudeMessage</code>	boolean	Optional. Whether the custom "thank you" message is displayed on all exit lobbies.
<code>customWaitingForChairMessage</code>	string (500)	Custom message displayed when waiting for a chair to join the conference. Only used if <code>useCustomWaitingForChairMessage</code> is <code>true</code> . Default: empty.
<code>useCustomOnlyVideoParticipantMessage</code>	boolean	Whether a custom message is displayed when a participant is the only (active) video participant. Default: <code>false</code> .
<code>customOnlyVideoParticipantMessage</code>	string (500)	Custom message displayed when a participant is the only (active) video participant. Only used if <code>useCustomOnlyVideoParticipantMessage</code> is <code>true</code> . Default: empty.
<code>useCustomConferenceEndingMessage</code>	boolean	Whether a custom message is displayed when the conference is about to end. Default: <code>false</code> .
<code>customConferenceEndingMessage</code>	string (100)	Custom message displayed when the conference is about to end. Only used if <code>useCustomConferenceEndingMessage</code> is <code>true</code> . Default: empty.
<code>metadata</code>	base64 (<= 512 bytes)	Client meta data. Default: zero length.
<code>unlockWithLastCall</code>	boolean	Whether the conference is unlocked when the participant leaves the conference. Default: <code>true</code> .
<code>guestControlLevel</code>	string	See <a href="#">Table 27 Control level enumerated type, page 31</a> . Either <code>controlNone</code> , <code>controlLocal</code> , or <code>controlConference</code> . Defines the level of control to which the guests in this conference are entitled. Default: <code>controlLocal</code> .
<code>chairControlLevel</code>	string	See <a href="#">Table 27 Control level enumerated type, page 31</a> . Either <code>controlNone</code> , <code>controlLocal</code> , or <code>controlConference</code> . Defines the level of control to which the chairpersons in this conference are entitled. Default: <code>controlConference</code> .
<code>optimizationProfile</code>	string	Sets the optimization profile for the conference, which defines how the conference reports far-end token values for endpoints. See <a href="#">Table 19 Optimization profiles enumerated type, page 29</a> . Default: <code>favorExperience</code> .

Table 72 flex.conference.create inputs (continued)

Parameter name	Type	Description
<code>useCustomMutedCanUnmuteMessage</code>	boolean	<code>true</code> enables a custom message that will be displayed to participants when their audio input has been muted on the TelePresence Server and they can unmute with *6.  Default: <code>true</code> .
<code>customMutedCanUnmuteMessage</code>	string (500)	The message displayed to participants when their audio has been muted on the TelePresence Server and they can unmute with *6. Will not be displayed if <code>useCustomMutedCanUnmuteMessage</code> is <code>false</code> .  Default: Empty.
<code>useCustomMutedCannotUnmuteMessage</code>	boolean	<code>true</code> enables a custom message that will be displayed when their audio has been muted on the TelePresence Server and they cannot unmute with *6.  Default: <code>true</code> .
<code>customMutedCannotUnmuteMessage</code>	string (500)	The message displayed to participants when their audio has been muted on the TelePresence Server and they cannot unmute with *6. Will not be displayed if <code>useCustomMutedCannotUnmuteMessage</code> is <code>false</code> .  Default: Empty.
<code>exitScreen</code>	boolean	Display a user friendly message when an endpoint is disconnected from a conference in an orderly manner. Default: <code>true</code> .
<code>useCustomConferenceEndedExitMessage</code>	boolean	Display custom message to a participant when they have been disconnected on Scheduled conference end, Conference deletion through API, Conference deletion through Web Interface and graceful Shutdown, if <code>exitScreen</code> is <code>true</code> . Default: <code>false</code> .
<code>customConferenceEndedExitMessage</code>	string (200)	Custom message displayed when a participant has been disconnected, if <code>useCustomConferenceEndedExitMessage</code> is <code>true</code> . Default: empty.
<code>useCustomParticipantDisconnectedExitMessage</code>	boolean	Display custom message to a participant when they have been disconnected using XCCP, API or Web interface or on No incoming media, if <code>exitScreen</code> is <code>true</code> . Default: <code>false</code> .
<code>customParticipantDisconnectedExitMessage</code>	string (200)	Custom message displayed when a participant has been disconnected, if <code>useCustomParticipantDisconnectedExitMessage</code> is <code>true</code> . Default: empty.
<code>customWelcomeScreenAudio</code>	string (80)	Optional. The url of a file to use as the voice prompt on the conference welcome screen.

Table 72 flex.conference.create inputs (continued)

Parameter name	Type	Description
<code>useCustomWelcomeScreenAudio</code>	boolean	Optional. Use the custom welcome screen voice prompt.
<code>customPINEntryAudio</code>	string (80)	Optional. The url of a file to use as the voice prompt on the PIN entry screen when the PIN is not optional.
<code>useCustomPINEntryAudio</code>	boolean	Optional. Use the custom PIN entry voice prompt.
<code>customOptionalPINEntryAudio</code>	string (80)	Optional. The url of a file to use as the voice prompt on the PIN entry screen when the PIN is optional
<code>useCustomOptionalPINEntryAudio</code>	boolean	Optional. Use the custom optional PIN entry voice prompt.
<code>customPINIncorrectAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt for incorrect PIN entry.
<code>useCustomPINIncorrectAudio</code>	boolean	Optional. Use the custom voice prompt for incorrect PIN
<code>customConferenceEndedExitAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt on the exit lobby on Scheduled conference end, Conference deletion API, Conference deletion Web Interface and Shutdown, if <code>exitScreen</code> is 1.
<code>useCustomConferenceEndedExitAudio</code>	boolean	Optional. Use the custom voice prompt for conference ending.
<code>customParticipantDisconnectedExitAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt on the exit lobby when a participant is disconnected using XCCP, API or Web interface or on No incoming media.
<code>useCustomParticipantDisconnectedExitAudio</code>	boolean	Optional. Use the custom voice prompt for participant disconnection.
<code>customPINFailedExitAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt on the exit lobby when a participant is disconnected for three failed PIN attempts.
<code>useCustomPINFailedExitAudio</code>	boolean	Optional. Use the custom voice prompt for participant disconnection on PIN failure.
<code>customWaitingForChairAudio</code>	string (url)	Optional. The url of a file to use as a custom voice prompt when waiting for the conference host.
<code>useCustomWaitingForChairAudio</code>	boolean	Optional. Use the custom voice prompt for waiting for the conference host.
<code>customOnlyParticipantAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt when there is only one participant in the conference.
<code>useCustomOnlyParticipantAudio</code>	boolean	Optional. Use the custom voice prompt for only participant.
<code>resourceOptimizationMode</code>	string (30)	Optional. Controls how resources in the conference are optimized. Default: <code>byParticipant</code> .
<code>customBackgroundImageURL</code>	string (80)	Optional. The url of a file to use as a custom background on lobby screens. Default: empty.

**Table 73** `flex.conference.create` returned data

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	Conference identifier assigned by the TelePresence Server. All subsequent invocations of commands to control or query this conference must use this identifier to reference it. See <a href="#">Identifiers and Client References, page 16</a> .
<code>conferenceReference</code>	string (50)	Conference client reference string. Returned if string length > 0. See <a href="#">Identifiers and Client References, page 16</a> .

`flex.conference.deletions.enumerate`

Enumerates deleted conferences. The enumeration returns conferences that have been newly deleted.

**Table 74** `flex.conference.deletions.enumerate` inputs

Parameter name	Type	Description
<code>cookie</code>	string (150)	Conference enumeration cookie. This field must be absent when starting an enumeration, and present (using the value returned by a previous invocation) when continuing an enumeration. Default: none.
<code>max</code>	integer (> 0)	Maximum number of conference deletion records to return in response. If <code>max</code> is not specified, as many records are returned as is possible.

**Table 75** `flex.conference.deletions.enumerate` returned data

Parameter name	Type	Description
<code>moreAvailable</code>	boolean	Whether there are more conference deletions to be enumerated.
<code>cookie</code>	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
<code>conferenceIDs</code>	array of identifiers	The identifiers of conferences that have been deleted. See <a href="#">Identifiers and Client References, page 16</a> .

`flex.conference.destroy`

Destroys the specified conference. No parameters are returned.

**Table 76** `flex.conference.destroy` inputs

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	<b>Required.</b> Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>allowExitScreen</code>	boolean	<b>Optional.</b> If <code>allowExitScreen</code> is <code>false</code> then, the <code>exitScreen</code> setting for the conference is overridden and set <code>false</code> . The exit lobby is not displayed and participant is disconnected immediately. Default: <code>False</code> .

`flex.conference.enumerate`

Enumerates conferences controlled by the TelePresence Server. The enumeration returns new conferences as well as conferences that have changed since the last invocation of the enumeration command. See [Enumeration, page](#)



## 21.

It is possible for two or more instances in succession of enumeration information structs returned for a particular conference to be identical. This may happen in the following circumstances:

- The summed token values are the same as before, but the distribution of tokens across participants has changed.
- Distribution of tokens was different for some period of time between successive invocations of the `flex.conference.enumerate` command.

**Table 77** `flex.conference.enumerate` inputs

Parameter name	Type	Description
<code>cookie</code>	string (150)	Conference enumeration cookie. This field must be absent when starting an enumeration, and present (using the value returned by a previous invocation) when continuing an enumeration. Default: none.
<code>max</code>	integer (> 0)	Maximum number of conference details to return in response. If <code>max</code> is not specified, as many records will be returned as is possible.

**Table 78** `flex.conference.enumerate` returned data

Parameter name	Type	Description
<code>moreAvailable</code>	boolean	Whether there are more conferences to be enumerated.
<code>cookie</code>	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
<code>conferences</code>	array of structs	Each member of the array is a struct representing a single conference. See <a href="#">Table 79 Conference information struct, page 66</a> . The array is returned empty if there are no conferences to enumerate.
<code>conferenceCreditsAllocated</code> (experimental only—do not use)	integer (>= 0)	Number of credits allocated to the conference (includes credits that are used and those that are reserved).
<code>conferenceCreditsRequired</code> (experimental only—do not use)	integer (>= 0)	Number of credits needed by the conference to provide the highest quality experience to all participants. May be greater than <code>maxMediaCredits</code> .
<code>conferenceCreditsUsed</code> (experimental only—do not use)	integer (>= 0)	Number of credits in use by this conference.

The enumerated conference information struct contains two sets of media resource values (tokens and credits).

- Configured: values set by configuration typically using this API.
- Allocated: the minimum of resource values corresponding to the capabilities of the near end and far end if a connection to the far end exists, or the configured values if media resources have been reserved and a call has not been established.

**Table 79 Conference information struct**

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>conferenceReference</code>	string (50)	Conference client reference string. Returned if string length > 0. See <a href="#">Identifiers and Client References, page 16</a> .
<code>conferenceName</code>	string (80)	Human readable label for the conference. Returned if string length > 0.
<code>locked</code>	boolean	Whether the conference is locked.
<code>active</code>	boolean	Whether the conference has started.
<code>terminating</code>	boolean	Whether the conference has been extended for the duration of the exit lobby.
<code>numParticipants</code>	integer (>= 0)	Number of participants connected to the conference, including participants with incoming calls that have not yet been established.
<code>callTokensConfiguredMainVideo</code>	integer (>= 0)	Number of tokens configured for main video summed over all participants connected to the conference.
<code>callTokensAllocatedMainVideo</code>	integer (>= 0)	Number of tokens allocated for main video summed over all participants connected to the conference.
<code>callTokensConfiguredExtendedVideo</code>	integer (>= 0)	Number of tokens configured for extended video summed over all participants connected to the conference.
<code>callTokensAllocatedExtendedVideo</code>	integer (>= 0)	Number of tokens allocated for extended video summed over all participants connected to the conference.
<code>callTokensConfiguredAudio</code>	integer (>= 0)	Number of tokens configured for audio summed over all participants connected to the conference.
<code>callTokensAllocatedAudio</code>	integer (>= 0)	Number of tokens allocated for audio summed over all participants connected to the conference.
<code>callQualityCanImproveMainVideo</code>	boolean	Whether at least one participant exists for which the number of far end main video tokens exceeds the number of configured main video tokens. The quality of the call can be improved by increasing the number of configured tokens.
<code>callQualityCanImproveExtendedVideo</code>	boolean	Whether at least one participant exists for which the number of far end extended video tokens exceeds the number of configured extended video tokens. The quality of the connection can be improved by increasing the number of configured tokens.
<code>callQualityCanImproveAudio</code>	boolean	Whether at least one participant exists for which the number of far end audio tokens exceeds the number of configured audio tokens. The quality of the connection can be improved by increasing the number of configured tokens.
<code>creditsConfigured</code>	integer (>= 0)	Number of configured credits summed over all participants connected to the conference.

**Table 79** Conference information struct (continued)

Parameter name	Type	Description
<code>creditsAllocated</code>	integer ( $\geq 0$ )	Number of allocated credits summed over all participants connected to the conference.
<code>presenterID</code>	string (50)	The identifier of the participant who is currently presenting to the conference. Not returned if no participant is presenting.
<code>importantID</code>	string (50)	The identifier of the participant who is currently marked as the conference's important participant. Not returned if no participant is marked as important. See <a href="#">flex.participant.setImportant</a> , page 102 and <a href="#">flex.participant.clearImportant</a> , page 83.

## flex.conference.getMetadata

Returns metadata associated with the specified conference. If a conference does not have metadata, 0 length metadata is returned.

**Table 80** flex.conference.getMetadata inputs

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	<b>Required.</b> Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References</a> , page 16.

**Table 81** flex.conference.getMetadata returned data

Parameter name	Type	Description
<code>metadata</code>	base64 ( $\leq 512$ bytes)	Client meta data.

## flex.conference.modify

Updates the specified parameters of the specified conference. No parameters are returned.

Settings that you do not specify remain as they were, unless implicitly affected by other settings that you supply. For example, if `duration` is specified, `durationUnlimited` is implicitly set to `false`.

The following pairs of parameters must be specified in compliance with the requirements stated in "[Unlimited Integers](#), page 20:

- `duration` and `durationUnlimited`
- `maxParticipants` and `maxParticipantsUnlimited`
- `conferenceMediaTokens` and `conferenceMediaTokensUnlimited`
- `conferenceMediaCredits` and `conferenceMediaCreditsUnlimited`

**Table 82** flex.conference.modify inputs

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	<b>Required.</b> Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References</a> , page 16.

Table 82 flex.conference.modify inputs (continued)

Parameter name	Type	Description
<code>conferenceReference</code>	string (50)	Conference client reference string. See <a href="#">Identifiers and Client References, page 16</a> .
<code>conferenceName</code>	string (80)	Human readable label for the conference.
<code>conferenceDescription</code>	string (500)	Human readable description of the conference. If set, the TelePresence Server uses this value for ActiveControl's conference description.
<code>URIs</code>	array of structs	<p>Each member of the array is a <a href="#">Table 33 Conference Connection Definitions, page 39</a> that defines conference URIs, associated access levels, and media resources.</p> <p>These replace all earlier conference URI entries. For URIs specified, all fields are set; unspecified optional fields are set to default values. If you supply an empty array, any previously defined conference URIs are deleted.</p> <p>You may supply a maximum of two conference URI structs. If you want two levels of access to the conference, for example for chairpersons and guests, you must create two structs in this array. Each struct in the array requires a unique URI.</p> <p><b>Note:</b> The URI does not need to be unique, if certain conditions are met. See <a href="#">URIs Do Not Need to be Unique, page 18</a></p>
<code>conferenceMediaTokens</code>	integer (>= 0)	Maximum number of media tokens that can be used for the conference. See <code>conferenceMediaTokensUnlimited</code> .
<code>conferenceMediaTokensUnlimited</code>	boolean	Whether no limit is defined for the number of media tokens that can be used for the conference.
<code>conferenceMediaCredits</code>	integer (>= 0)	Maximum number of media credits that can be used for the conference. See <code>conferenceMediaCreditsUnlimited</code> .
<code>conferenceMediaCreditsUnlimited</code>	boolean	Whether no limit is defined for the number of media credits that can be used for the conference.
<code>waitForChair</code>	boolean	Whether callers must wait for a chair to join the conference.
<code>disconnectOnChairExit</code>	boolean	Whether callers are disconnected when the last chair leaves the conference.
<code>terminateWithLastCall</code>	boolean	Whether the conference is destroyed with the last call.
<code>locked</code>	boolean	Whether the conference is locked causing incoming calls to be rejected.

Table 82 flex.conference.modify inputs (continued)

Parameter name	Type	Description
<code>duration</code>	integer ( $\geq 0$ )	Conference duration (in seconds) measured from the start time. If the conference is due to end in less than 120 seconds, participants are notified that it is about to end, as described in <a href="#">Conference send warning</a> . It is an error to set the duration to a value that requires it to have ended in the past. See <code>durationUnlimited</code> .
<code>durationUnlimited</code>	boolean	Whether an unlimited duration is assigned for the conference.
<code>billingCode</code>	string (80)	Billing code string.
<code>callAttributes</code>	struct	This <a href="#">Call Attributes Struct, page 34</a> defines the conference default call attributes. See <a href="#">How Call Attributes are Derived, page 34</a> .
<code>participantMediaResources</code>	struct	This <a href="#">Participant Media Resources Struct, page 33</a> struct defines the conference default participant media resource configuration.  These settings can be overridden by those defined at the <code>Conference URI</code> or participant level. All members of the struct are changed; unspecified optional fields are set to their default values.
<code>maxParticipants</code>	integer ( $\geq 0$ )	Maximum number of participants that can connect to this conference. See <code>maxParticipantsUnlimited</code> .
<code>maxParticipantsUnlimited</code>	boolean	Whether an unlimited number of participants are allowed to connect to this conference. See <a href="#">"Unlimited" Integers, page 20</a> .
<code>voiceSwitchingSensitivity</code>	integer (0-100)	Voice switching sensitivity. Default: 60.
<code>welcomeScreen</code>	boolean	Whether a welcome screen message is displayed for 5 seconds when a call joins the conference. See <code>welcomeScreenMessage</code> for the contents of the message.
<code>welcomeScreenMessage</code>	string (500)	Welcome screen message for this conference. If this message is empty, the conference name is displayed as the welcome screen message.  If <code>conferenceDescription</code> is not set, the TelePresence Server uses this value for ActiveControl's conference description.
<code>useCustomPINEntryMessage</code>	boolean	Whether a custom message is displayed in the PIN entry screen.
<code>customPINEntryMessage</code>	string (200)	Custom message for PIN entry. Only used if <code>useCustomPINEntryMessage</code> is true.
<code>useCustomOptionalPINEntryMessage</code>	boolean	Display custom message at shared PIN entry screen when conference can be entered without a PIN

Table 82 flex.conference.modify inputs (continued)

Parameter name	Type	Description
<code>customOptionalPINEntryMessage</code>	string (200)	Custom message for shared PIN entry screen when conference can be entered without a PIN, if <code>useCustomOptionalPINEntryMessage</code> is true.
<code>useCustomPINIncorrectMessage</code>	boolean	Whether a custom warning message is displayed in the PIN entry screen after an incorrect PIN has been entered.
<code>customPINIncorrectMessage</code>	string (100)	Custom warning message for incorrect PIN entry. Only used if <code>useCustomPINIncorrectMessage</code> is true.
<code>useCustomWaitingForChairMessage</code>	boolean	Whether a custom message is displayed when waiting for a chair to join the conference.
<code>customWaitingForChairMessage</code>	string (500)	Custom message displayed when waiting for a chair to join the conference. Only used if <code>useCustomWaitingForChairMessage</code> is true.
<code>customPINEntryFailedMessage</code>	string (200)	Optional. Custom message displayed on the exit lobby if a participant is disconnected for failing to enter the PIN 3 times.
<code>useCustomPINEntryFailedMessage</code>	boolean	Optional. Whether the custom message is displayed for the PIN failure exit lobby.
<code>customDisconnectPlatitudeMessage</code>	string (200)	Optional. Custom "thank you" message displayed on all exit lobbies.
<code>useCustomDisconnectPlatitudeMessage</code>	boolean	Optional. Whether the custom "thank you" message is displayed on all exit lobbies.
<code>useCustomOnlyVideoParticipantMessage</code>	boolean	Whether a custom message is displayed when a participant is the only (active) video participant.
<code>customOnlyVideoParticipantMessage</code>	string (500)	Custom message displayed when a participant is the only (active) video participant. Only used if <code>useCustomOnlyVideoParticipantMessage</code> is true.
<code>useCustomConferenceEndingMessage</code>	boolean	Whether a custom message is displayed when the conference is about to end.
<code>customConferenceEndingMessage</code>	string (100)	Custom message displayed when the conference is about to end. Only used if <code>useCustomConferenceEndingMessage</code> is true.
<code>metadata</code>	base64 (<= 512 bytes)	Client metadata.
<code>unlockWithLastCall</code>	boolean	Whether the conference is unlocked when the participant leaves the conference.
<code>guestControlLevel</code>	string	See <a href="#">Table 27 Control level enumerated type, page 31</a> . Either <code>controlNone</code> , <code>controlLocal</code> , or <code>controlConference</code> . Defines the level of control to which the guests in this conference are entitled.

Table 82 flex.conference.modify inputs (continued)

Parameter name	Type	Description
<code>chairControlLevel</code>	string	See <a href="#">Table 27 Control level enumerated type, page 31</a> . Either <code>controlNone</code> , <code>controlLocal</code> , or <code>controlConference</code> . Defines the level of control to which the chairpersons in this conference are entitled.
<code>optimizationProfile</code>	string	Sets the optimization profile for the conference, which defines how the conference reports far-end token values for endpoints. See <a href="#">Table 19 Optimization profiles enumerated type, page 29</a> .  Default: <code>favorExperience</code> .
<code>useCustomMutedCanUnmuteMessage</code>	boolean	<code>true</code> enables a custom message that will be displayed to participants when their audio input has been muted on the TelePresence Server and they can unmute with *6.  Default: <code>true</code> .
<code>customMutedCanUnmuteMessage</code>	string (500)	The message displayed to participants when their audio has been muted on the TelePresence Server and they can unmute with *6. Will not be displayed if <code>useCustomMutedCanUnmuteMessage</code> is <code>false</code> .  Default: Empty.
<code>useCustomMutedCannotUnmuteMessage</code>	boolean	<code>true</code> enables a custom message that will be displayed when their audio has been muted on the TelePresence Server and they cannot unmute with *6.  Default: <code>true</code> .
<code>customMutedCannotUnmuteMessage</code>	string (500)	The message displayed to participants when their audio has been muted on the TelePresence Server and they cannot unmute with *6. Will not be displayed if <code>useCustomMutedCannotUnmuteMessage</code> is <code>false</code> .  Default: Empty.
<code>exitScreen</code>	boolean	Display a user friendly message when an endpoint is disconnected from a conference in an orderly manner. Default: <code>true</code> .
<code>useCustomConferenceEndedExitMessage</code>	boolean	Display custom message to a participant when they have been disconnected on Scheduled conference end, Conference deletion through API, Conference deletion through Web Interface and graceful Shutdown, if <code>exitScreen</code> is <code>true</code> . Default: <code>false</code>
<code>customConferenceEndedExitMessage</code>	string (200)	Custom message displayed when a participant has been disconnected, if <code>useCustomConferenceEndedExitMessage</code> is <code>true</code> . Default: empty

**Table 82** flex.conference.modify inputs (continued)

Parameter name	Type	Description
useCustomParticipantDisconnectedExitMessage	boolean	Display custom message to a participant when they have been disconnected using XCCP, API or Web interface or on No incoming media, if exitScreen is true. Default: false
customParticipantDisconnectedExitMessage	string (200)	Custom message displayed when a participant has been disconnected, if useCustomParticipantDisconnectedExitMessage is true. Default: empty

## flex.conference.query

Returns the parameters of a specified conference.

### Custom Lobby Screen Disclaimer

The custom lobby screen background feature in TelePresence Server is provided as a preview feature and is not intended for use in production environments. Use of the custom lobby screen background feature is subject to the software license and limited warranty [http://www.cisco.com/c/en/us/td/docs/general/warranty/English/EU1KEN\\_.html](http://www.cisco.com/c/en/us/td/docs/general/warranty/English/EU1KEN_.html) for TelePresence Server. Cisco reserves the right to disable the custom lobby screen background feature at any time without notice. Cisco Technical Support will provide limited assistance to customers who wish to use the custom lobby screen background feature.

**Table 83** flex.conference.query inputs

Parameter name	Type	Description
conferenceID	string (50)	<b>Required.</b> Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .

**Table 84** flex.conference.query returned data

Parameter name	Type	Description
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
URIs	array of structs	Each member of the array is a <a href="#">Table 33 Conference Connection Definitions, page 39</a> defining a unique conference URI and its associated access levels and media tokens.  The array contains a maximum of two conference URI structs. These can be used to allow two levels of access to the conference, for example for chairpersons and guests. Each struct in the array requires a unique URI.
conferenceReference	string (50)	Conference client reference string. Returned if string length > 0. See <a href="#">Identifiers and Client References, page 16</a> .



Table 84 flex.conference.query returned data (continued)

Parameter name	Type	Description
<code>conferenceName</code>	string (80)	Human readable label for the conference. Returned if string length > 0.
<code>conferenceDescription</code>	string (500)	Human readable description of the conference. If set, the TelePresence Server uses this value for ActiveControl's conference description. Not returned if it has not been set.
<code>conferenceMediaTokens</code>	integer (>= 0)	Maximum number of media tokens that can be used for the conference. See <code>conferenceMediaTokensUnlimited</code> . Not returned if it has not been set.
<code>conferenceMediaTokensUnlimited</code>	boolean	Whether no limit is defined for the number of media tokens that can be used for the conference. Not returned if it has not been set. See " <a href="#">Unlimited Integers</a> , page 20.
<code>conferenceMediaCredits</code>	integer (>= 0)	Maximum number of media credits that can be used for the conference. See <code>conferenceMediaCreditsUnlimited</code> . Not returned if it has not been set.
<code>conferenceMediaCreditsUnlimited</code>	boolean	Whether no limit is defined for the number of media credits that can be used for the conference. Not returned if it has not been set. See " <a href="#">Unlimited Integers</a> , page 20.
<code>duration</code>	integer (>= 0)	Conference duration (in seconds) measured from the start time. If the conference is due to end in less than 120 seconds, participants are notified that it is about to end, as described in <a href="#">Conference send warning</a> . Not returned if it has not been set. See <code>durationUnlimited</code> .
<code>durationUnlimited</code>	boolean	Whether an unlimited duration is assigned for the conference. Not returned if it has not been set. See " <a href="#">Unlimited Integers</a> , page 20.
<code>billingCode</code>	string (80)	Billing code string. Not returned if it has not been set.
<code>maxParticipants</code>	integer (>= 0)	Maximum number of participants that can connect to this conference. Not returned if it has not been set. See <code>maxParticipantsUnlimited</code> .
<code>maxParticipantsUnlimited</code>	boolean	Whether an unlimited number of participants are allowed to connect to this conference. Not returned if it has not been set. See " <a href="#">Unlimited Integers</a> , page 20.
<code>waitForChair</code>	boolean	Whether callers must wait for a chair to join the conference.
<code>disconnectOnChairExit</code>	boolean	Whether callers are disconnected when the last chair leaves the conference.

Table 84 flex.conference.query returned data (continued)

Parameter name	Type	Description
<code>terminateWithLastCall</code>	boolean	Whether the conference is destroyed with the last call.
<code>locked</code>	boolean	Whether the conference is locked so that only outgoing calls are permitted.
<code>startTime</code>	integer (>= 0)	Number of seconds after which to start the conference.
<code>callAttributes</code>	struct	This <a href="#">Call Attributes Struct, page 34</a> defines the conference default call attributes. See <a href="#">How Call Attributes are Derived, page 34</a> .
<code>voiceSwitchingSensitivity</code>	integer (0-100)	Voice switching sensitivity.
<code>participantMediaResources</code>	struct	This <a href="#">Participant Media Resources Struct, page 33</a> defines the conference default participant media resource configuration.  These settings can be over-ridden by those defined at the <code>Conference URI</code> or participant level.
<code>welcomeScreen</code>	boolean	Whether a welcome screen message is displayed for 5 seconds when a call joins the conference. See <code>welcomeScreenMessage</code> for contents of the message.
<code>welcomeScreenMessage</code>	string (500)	Welcome screen message for this conference. If this message is empty, the conference name is displayed as the welcome screen message.  If <code>conferenceDescription</code> is not set, the TelePresence Server uses this value for ActiveControl's conference description.
<code>useCustomPINEntryMessage</code>	boolean	Whether a custom message is displayed in the PIN entry screen.
<code>customPINEntryMessage</code>	string (200)	Custom message for PIN entry. Only used if <code>useCustomPINEntryMessage</code> is true.
<code>useCustomOptionalPINEntryMessage</code>	boolean	Display custom message at shared PIN entry screen when conference can be entered without a PIN
<code>customOptionalPINEntryMessage</code>	string (200)	Custom message for shared PIN entry screen when conference can be entered without a PIN, if <code>useCustomOptionalPINEntryMessage</code> is true.
<code>useCustomPINIncorrectMessage</code>	boolean	Whether a custom warning message is displayed in the PIN entry screen after an incorrect PIN has been entered.
<code>customPINIncorrectMessage</code>	string (100)	Custom warning message for incorrect PIN entry, Only used if <code>useCustomPINIncorrectMessage</code> is true.

Table 84 flex.conference.query returned data (continued)

Parameter name	Type	Description
<code>useCustomWaitingForChairMessage</code>	boolean	Whether a custom message is displayed when waiting for a chair to join the conference.
<code>customWaitingForChairMessage</code>	string (500)	Custom message displayed when waiting for a chair to join the conference. Only used if <code>useCustomWaitingForChairMessage</code> is true.
<code>useCustomOnlyVideoParticipantMessage</code>	boolean	Whether a custom message is displayed when the participant is the only (active) video participant.
<code>customOnlyVideoParticipantMessage</code>	string (500)	Custom message displayed when the participant is the only (active) video participant. Only used if <code>useCustomOnlyVideoParticipantMessage</code> is true.
<code>useCustomConferenceEndingMessage</code>	boolean	Whether a custom message is displayed when the conference is about to end.
<code>customConferenceEndingMessage</code>	string (100)	Custom message displayed when the conference is about to end. Only used if <code>useCustomConferenceEndingMessage</code> is true.
<code>hasMetadata</code>	boolean	Whether the conference has non-zero-length metadata.
<code>unlockWithLastCall</code>	boolean	Whether the conference is unlocked when the last participant leaves the conference.
<code>guestControlLevel</code>	string	See <a href="#">Table 27 Control level enumerated type, page 31</a> . Either <code>controlNone</code> , <code>controlLocal</code> , or <code>controlConference</code> . Defines the level of control to which the guests in this conference are entitled.
<code>chairControlLevel</code>	string	See <a href="#">Table 27 Control level enumerated type, page 31</a> . Either <code>controlNone</code> , <code>controlLocal</code> , or <code>controlConference</code> . Defines the level of control to which the chairpersons in this conference are entitled.
<code>optimizationProfile</code>	string	The optimization profile for the conference, which defines how the conference reports far-end token values for endpoints. See <a href="#">Table 19 Optimization profiles enumerated type, page 29</a> .
<code>useCustomMutedCanUnmuteMessage</code>	boolean	<code>true</code> means that a custom message can be displayed to participants when their audio input has been muted on the TelePresence Server and they can unmute with *6.
<code>customMutedCanUnmuteMessage</code>	string (500)	The message that will be displayed to participants when their audio input has been muted on the TelePresence Server and they can unmute with *6. Will not be displayed if <code>useCustomMutedCanUnmuteMessage</code> is false.

Table 84 flex.conference.query returned data (continued)

Parameter name	Type	Description
<code>useCustomMutedCannotUnmuteMessage</code>	boolean	<code>true</code> means that a custom message can be displayed to participants when their audio input has been muted on the TelePresence Server and they cannot unmute with *6.
<code>customMutedCannotUnmuteMessage</code>	string (500)	The message that will be displayed to participants when their audio input has been muted on the TelePresence Server and they cannot unmute with *6. Will not be displayed if <code>useCustomMutedCannotUnmuteMessage</code> is <code>false</code> .
<code>exitScreen</code>	boolean	Display a user friendly message when an endpoint is disconnected from a conference in an orderly manner. Default: <code>true</code>
<code>useCustomConferenceEndedExitMessage</code>	boolean	Display custom message to a participant when they have been disconnected on Scheduled conference end, Conference deletion through API, Conference deletion through Web Interface and graceful Shutdown, if <code>exitScreen</code> is <code>true</code> . Default: <code>false</code>
<code>customConferenceEndedExitMessage</code>	string (200)	Custom message displayed when a participant has been disconnected, if <code>useCustomConferenceEndedExitMessage</code> is <code>true</code> . Default: empty
<code>useCustomParticipantDisconnectedExitMessage</code>	boolean	Display custom message to a participant when they have been disconnected using XCCP, API or Web interface or on No incoming media, if <code>exitScreen</code> is <code>true</code> . Default: <code>false</code>
<code>customParticipantDisconnectedExitMessage</code>	string (200)	Custom message displayed when a participant has been disconnected, if <code>useCustomParticipantDisconnectedExitMessage</code> is <code>true</code> . Default: empty
<code>customPINEntryFailedMessage</code>	string (200)	Optional. Custom message displayed on the exit lobby if a participant is disconnected for failing to enter the PIN 3 times.
<code>useCustomPINEntryFailedMessage</code>	boolean	Optional. Whether the custom message is displayed for the PIN failure exit lobby.
<code>customDisconnectPlatitudeMessage</code>	string (200)	Optional. Custom "thank you" message displayed on all exit lobbies.
<code>useCustomDisconnectPlatitudeMessage</code>	boolean	Optional. Whether the custom "thank you" message is displayed on all exit lobbies.
<code>customWelcomeScreenAudio</code>	string (80)	Optional. The url of a file to use as the voice prompt on the conference welcome screen.
<code>useCustomWelcomeScreenAudio</code>	boolean	Optional. Use the custom welcome screen voice prompt.

**Table 84 flex.conference.query returned data (continued)**

Parameter name	Type	Description
<code>customPINEntryAudio</code>	string (80)	Optional. The url of a file to use as the voice prompt on the PIN entry screen when the PIN is not optional.
<code>useCustomPINEntryAudio</code>	boolean	Optional. Use the custom PIN entry voice prompt.
<code>customOptionalPINEntryAudio</code>	string (80)	Optional. The url of a file to use as the voice prompt on the PIN entry screen when the PIN is optional
<code>useCustomOptionalPINEntryAudio</code>	boolean	Optional. Use the custom optional PIN entry voice prompt.
<code>customPINIncorrectAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt for incorrect PIN entry.
<code>useCustomPINIncorrectAudio</code>	boolean	Optional. Use the custom voice prompt for incorrect PIN
<code>customConferenceEndedExitAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt on the exit lobby on Scheduled conference end, Conference deletion API, Conference deletion Web Interface and Shutdown, if exitScreen is 1.
<code>useCustomConferenceEndedExitAudio</code>	boolean	Optional. Use the custom voice prompt for conference ending.
<code>customParticipantDisconnectedExitAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt on the exit lobby when a participant is disconnected using XCCP, API or Web interface or on No incoming media.
<code>useCustomParticipantDisconnectedExitAudio</code>	boolean	Optional. Use the custom voice prompt for participant disconnection.
<code>customPINFailedExitAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt on the exit lobby when a participant is disconnected for three failed PIN attempts.
<code>useCustomPINFailedExitAudio</code>	boolean	Optional. Use the custom voice prompt for participant disconnection on PIN failure.
<code>customWaitingForChairAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt when waiting for the conference host.
<code>useCustomWaitingForChairAudio</code>	boolean	Optional. Use the custom voice prompt for waiting for the conference host.
<code>customOnlyParticipantAudio</code>	string (80)	Optional. The url of a file to use as a custom voice prompt when there is only one participant in the conference.
<code>useCustomOnlyParticipantAudio</code>	boolean	Optional. Use the custom voice prompt for only participant.
<code>customBackgroundImageURL</code>	string (url 80)	Optional. The url of a file to use as a custom background on lobby screens. Default: empty.

## flex.conference.sendUserMessage

Sends a message to all participants in the conference. For multi-call participants, the message is sent to the call in the center.

**Table 85** flex.conference.sendUserMessage inputs

Parameter name	Type	Description
conferenceID	string (50)	<b>Required.</b> Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
message	string (100)	<b>Required.</b> Message to display.
duration	integer (>0)	Duration in seconds for the message display on the endpoint. The TelePresence Server will accept 0 but the behavior is undefined in this case.  Default: 30 seconds.

### Message display behavior

Messages display in the following priority order:

1. Custom on-screen messages
2. Conference ending warning
3. \*6 muting
4. External muting

**Note:** that the duration remaining of a higher priority active message determines whether you see the lower priority message, such as \*6 muting, which may only display briefly, if at all.

However, a higher priority message sent to a participant or conference will always override a lower priority message if one is being displayed.

## flex.conference.sendWarning

Sends a warning to all participants in the specified conference that the conference is about to end.

If possible, a participant is notified that the conference is about to end using an appropriate out-of-band protocol. Otherwise, a message is rendered on the participant screen.

**Table 86** flex.conference.sendWarning inputs

Parameter name	Type	Description
conferenceID	string (50)	<b>Required.</b> Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
secondsRemaining	integer (>=0)	Additional information for the warning, namely the amount of time remaining until the conference is expected to terminate. Some endpoints are capable of receiving and using this information. Setting this value will <b>not</b> result in termination of the conference after the specified amount of time. Default: 120 seconds if the conference has no defined ending time. There is no default for conferences with a finite duration.

## flex.conference.status

Returns the status of the specified conference.

**Table 87** `flex.conference.status` inputs

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	<b>Required.</b> Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .

**Table 88** `flex.conference.status` returned data

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>locked</code>	boolean	Whether the conference is locked.
<code>numParticipants</code>	integer (>= 0)	Number of participants connected to the conference, including participants with incoming calls that have not yet been established.
<code>numScreens</code>	integer (>= 0)	Number of screens connected to the conference.
<code>configuredMediaTokens</code>	integer	Sum of configured media tokens for all calls connected to the conference.
<code>configuredMediaCredits</code>	integer	Sum of configured media credits for all calls connected to the conference.
<code>startTime</code>	integer	Amount of time, measured in seconds, between now and the conference start time. This value is < 0 for conferences that are currently active.
<code>active</code>	boolean	Whether the conference has started.
<code>presenterID</code>	string (50)	Identifier of the participant who is currently presenting to the conference. Not returned if no participant is presenting.
<code>conferenceReference</code>	string (50)	Conference client reference string. Returned if string length > 0. See <a href="#">Identifiers and Client References, page 16</a> .
<code>endTime</code>	integer	Amount of time, measured in seconds, until the conference is due to end. Only returned if conference duration is limited.
<code>conferenceCreditsAllocated</code> (experimental only—do not use)	integer (>= 0)	Number of credits allocated to the conference (includes credits that are used and those that are reserved).
<code>conferenceCreditsRequired</code> (experimental only—do not use)	integer (>= 0)	Number of credits needed by the conference to provide the highest quality experience to all participants. May be greater than <code>maxMediaCredits</code> .
<code>conferenceCreditsUsed</code> (experimental only—do not use)	integer (>= 0)	Number of credits in use by this conference.

`flex.licenseMode.modify`

**Note:** These commands and parameters are for internal use by Cisco only.

Only supported when in Multiparty licensed mode (use `system.info` to query).

**Table 89** `flex.licenseMode.modify` inputs

Parameter name	Type	Description
<code>newLicenseMode</code>	String (7)	One of <code>trust</code> or <code>key</code> . TelePresence Server boots into Screen licensed mode ( <code>key</code> ) by default.
<code>version</code>	Integer	Version to be used.

**Table 90** `flex.licenseMode.modify` returned data

Parameter name	Type	Description
<code>trustSecret</code>	base64	Trust secret.

`flex.licenseMode.verify`

**Note:** These commands and parameters are for internal use by Cisco only.

Only supported when in Multiparty licensed mode (use `system.info` to query).

**Table 91** `flex.licenseMode.verify` inputs

Parameter name	Type	Description
No additional parameters required.		

**Table 92** `flex.licenseMode.verify` returned data

Parameter name	Type	Description
None		

`flex.participant.advanced.enumerate`

Enumerates participants. This command is an alternative for `flex.participant.enumerate`. `flex.participant.enumerate` is still accepted, but you should only use one of these methods for participant enumeration.

See [Enumeration, page 21](#) and [flex.participant.enumerate, page 87](#).

**Table 93** `flex.participant.advanced.enumerate` inputs

Parameter name	Type	Description
<code>cookie</code>	string (150)	Participant enumeration cookie. This field must be absent at the start of the enumeration, and present (using the value returned by a previous invocation) when continuing an enumeration. Default: none.
<code>max</code>	integer (> 0)	Maximum number of participant details to return in response. If <code>max</code> is not specified, as many records will be returned as is possible.
<code>conferenceID</code>	string (50)	Enumerates only participants in the specified conference. Can only be supplied when <code>cookie</code> is absent. Enumeration for non-existent conferences will fail. See <a href="#">Identifiers and Client References, page 16</a> . Default: none.



**Table 94** `flex.participant.advanced.enumerate` returned data

Parameter name	Type	Description
<code>moreAvailable</code>	boolean	Whether there are more participants to be enumerated.
<code>cookie</code>	string (150)	Enumeration cookie that must be returned in the next invocation to continue the enumeration. See <a href="#">Enumeration, page 21</a> .
<code>participants</code>	array of structs	Each member of the array is a struct defining a participant. See <a href="#">Table 95 Participant information struct members, page 81</a> . This array may be empty.

**Table 95** Participant information struct members

Parameter name	Type	Description
<code>participantID</code>	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>participantReference</code>	string (50)	Client reference string. See <a href="#">Identifiers and Client References, page 16</a> . Absent if empty.
<code>displayName</code>	string (80)	The current display name for this participant. Absent if empty.
<code>conferenceID</code>	string (50)	Conference to which the participant is connected. See <a href="#">Identifiers and Client References, page 16</a> .
<code>accessLevel</code>	string	Access level granted to the participant. See <a href="#">Table 7 Access level enumerated type, page 27</a> .  <b>Note:</b> Incoming calls are reported as <code>unknown</code> until they have been authorized (reach the lobby screen).
<code>connectionState</code>	string	One of <code>disconnected</code> , <code>connecting</code> , <code>connected</code> , or <code>onHold</code> . See <a href="#">Table 22 Participant connection state enumerated type, page 30</a> .
<code>calls</code>	array of structs	Each member of this array is a struct that defines one call associated with the participant. See <a href="#">Table 96 Participant call information struct members, page 82</a> .  If a call is absent, the array member is empty.  The array position of the call information struct matches the position of the corresponding address struct in the <code>addresses</code> array.
<code>addresses</code>	array of structs	Each member of this array is a struct that contains either the address or the URI for one of the calls associated with the participant.  See <a href="#">Table 97 Participant address struct members, page 82</a> .  The position in the array matches that of the associated call in the <code>calls</code> array.
<code>encryptionStatus</code>	struct	An overview of the participant's encryption status. Each struct member represents a channel, or category of channels, that can be encrypted for this participant. The value of each member is one of the encryption status enumerated types. See <a href="#">Table 98 encryptionStatus struct members, page 82</a> and <a href="#">Table 23 Encryption status enumerated type, page 30</a> .

**Table 95 Participant information struct members (continued)**

Parameter name	Type	Description
<code>layout</code>	string	The display layout that is currently shown on the participant's endpoint. One of <code>layoutSingle</code> , <code>layoutActivepresence</code> , <code>layoutProminent</code> , or <code>layoutEqual</code> . Not returned if there are no calls connected for this participant.  See <a href="#">Table 10 Single screen layout enumerated type, page 27</a> or <a href="#">Table 11 Multi-screen layout enumerated type, page 27</a> , depending on the type of endpoint.
<code>mediaStatus</code>	struct	An overview of the participant's media status. Each struct member represents a media channel, or category of media channels, that can be negotiated for this participant. The value of each member is one of the media status enumerated types. See <a href="#">Table 99 mediaStatus struct members, page 83</a> and <a href="#">Enumerated Types, page 26</a> .

**Table 96 Participant call information struct members**

Parameter name	Type	Description
<code>callID</code>	string (50)	Call identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>incoming</code>	boolean	Direction of the call: <code>true</code> indicates incoming; <code>false</code> indicates outgoing.
<code>address</code>	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.
<code>protocol</code>	string	The call control protocol used in this call. One of <code>h323</code> or <code>sip</code> . See <a href="#">Enumerated Types, page 26</a> .
<code>rxBandwidth</code>	integer	
<code>txBandwidth</code>	integer	

**Table 97 Participant address struct members**

Parameter name	Type	Description
<code>URI</code>	string (80)	URI used or to be used by the endpoint to connect to the conference. Incoming calls only, not returned for outgoing calls.  Only returned if it is not empty.
<code>remoteAddress</code>	string (80)	Remote address specified in <a href="#">flex.participant.create, page 84</a> to call out to the endpoint. Outgoing calls only, not returned for incoming calls.  Only returned if it is not empty.

**Table 98 encryptionStatus struct members**

Parameter name	Type	Description
<code>callProtocol</code>	string	The encryption status of the call control protocol. One of <code>unknown</code> , <code>encrypted</code> , <code>mixed</code> , or <code>unencrypted</code> . See <a href="#">Table 23 Encryption status enumerated type, page 30</a> .

**Table 98** encryptionStatus struct members (continued)

Parameter name	Type	Description
audio	string	The encryption status of this participant's audio channel(s)
mainVideo	string	The encryption status of this participant's video channel(s)
extendedVideo	string	The encryption status of this participant's content channel
cccp	string	The encryption status of the Cisco Conference Control Protocol (CCCP).
activeControl	string	The encryption status of the ActiveControl signaling channel.

**Table 99** mediaStatus struct members

Parameter name	Type	Description
audioRx	string	Media status of the audio channel(s) received from this participant. One of notNegotiated, inUse, notInUse, Or muted. See <a href="#">Enumerated Types, page 26</a> .
audioTx	string	Media status of the audio channel(s) transmitted to this participant.
videoRx	string	Media status of the video channel(s) received from this participant.
videoTx	string	Media status of the video channel(s) transmitted to this participant.
extendedRx	string	Media status of the content channel(s) received from this participant.
extendedTx	string	Media status of the content channel(s) transmitted to this participant.

## flex.participant.call.disconnect

Disconnects an incoming call that is connected through a participant conference URI.

Outgoing calls cannot be disconnected. To change the destination of an outgoing call, the participant must be destroyed and recreated with the new address.

**Table 100** flex.participant.call.disconnect inputs

Parameter name	Type	Description
participantID	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
position	integer (>= 0)	<b>Required.</b> Relative position of the endpoint in the group. 0 represents the leftmost screen from a viewing position.

This command fails under the following circumstances:

- The participant call has not connected through a participant conference URI.
- The participant call is an outgoing call.
- The position value is invalid for the participant.

## flex.participant.clearImportant

Removes the designation of the specified participant as the important participant.

**Table 101** `flex.participant.clearImportant` inputs

Parameter name	Type	Description
<code>participantID</code>	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .

## `flex.participant.create`

Creates single- or multi-call participants associated with the specified conference.

If the command is successful, media resources (tokens and credits) are reserved for the new participant. See [Media Reservation, page 21](#)

**Note:** The token requirements for a call cannot be known prior to instantiation of the call, so no checks are made on `flex.participant.create` or `flex.participant.modify` to determine if the call will have adequate resources. The client is therefore responsible for ensuring that the call has adequate resources.

The following circumstances can cause this command to fail:

- The `audioIndex` and `contentIndex` values are invalid
- The rules for [Participant Call Definition Struct, page 40](#) are not met
- `cascadeRole` is not `cascadeNone` and more than one call is defined in the `calls` array

**Table 102** `flex.participant.create` inputs

Parameter name	Type	Description
<code>conferenceID</code>	string (50)	<b>Required.</b> Conference identifier. See <a href="#">Identifiers and Client References, page 16</a> .
<code>calls</code>	array of structs	<b>Required.</b> Each member of the array is a <a href="#">Participant Call Definition Struct, page 40</a> which specifies a call for this participant; you must supply a minimum of 1 call definition struct and may supply up to 4. For multi-call participants, the array position of the struct corresponds to the call's physical location with respect to other calls. For example, the array positions of a three-call participant correspond to physical locations as follows: position 0 is left, 1 is center, and 2 is right.  For endpoints that automatically negotiate extra screens (such as a T3), you need only specify the call at position 0 and the remaining calls will be added as they are negotiated.
<code>participantReference</code>	string (50)	Client reference string. See <a href="#">Identifiers and Client References, page 16</a> . Default: empty.

Table 102 flex.participant.create inputs (continued)

Parameter name	Type	Description
<code>PIN</code>	string (40)	Numeric PIN this participant will use when connecting to conference URIs. Participants only need to supply a PIN when calling in to a PIN-protected URI.  A PIN is never requested when the TelePresence Server calls out to an endpoint. If a PIN is supplied to this call when it is not required (because all the calls are outgoing), then the TelePresence Server returns fault code 102.  Default: empty.  <b>Deprecated:</b> please use <code>PINs</code> instead
<code>PINs</code>	array of <a href="#">Participant PIN Definition</a> , page 41	List of PINs for this participant, if any of its calls are incoming. Maximum of 2.
<code>callAttributes</code>	struct	See <a href="#">Call Attributes Struct</a> , page 34 for details of struct members. The settings defined in this struct override the conference's default call attribute settings. See <a href="#">How Call Attributes are Derived</a> , page 34. Default: inherits conference default call attributes.
<code>participantMediaResources</code>	struct	See <a href="#">Participant Media Resources Struct</a> , page 33 for details of struct members. The settings defined in this struct override the conference's default participant media resource configuration. Default: inherits conference default participant media resource configuration.
<code>camerasCrossed</code>	boolean	Whether cameras in the group of endpoints specified by <code>calls</code> are crossed. Ignored if the calls array length = 1. Default: <code>false</code> .
<code>audioIndex</code>	integer (>= 0)	Position in the <code>calls</code> array of the call that will receive audio. The position must exist in the <code>calls</code> array. First position is 0. Ignored if the <code>calls</code> array length = 1. Default: 0.
<code>contentIndex</code>	integer (>= 0)	Position in the <code>calls</code> array of the call that will receive content. The position must exist in the <code>calls</code> array. First position is 0. Ignored if the <code>calls</code> array length = 1. Default: 0.
<code>displayName</code>	string (80)	Configured display name for the endpoint. This overrides the endpoint display name setting. Default: empty.
<code>dtmf</code>	string (127)	Valid <a href="#">DTMF</a> , page 23 characters in a sequence that is sent to the call nominated by the <code>audioIndex</code> . This sequence is only sent when dialing out and is not sent for incoming calls. Default: empty.
<code>callerName</code>	string (80)	Calling name seen by the endpoint. Not used for incoming calls. Default: empty.
<code>callerAddress</code>	string (80)	Calling address seen by the endpoint. Not used for incoming calls. Default: empty.
<code>cascadeRole</code>	string	One of <code>cascadeNone</code> , <code>cascadeMaster</code> , or <code>cascadeSlave</code> . See <a href="#">Table 18 Cascade roles enumerated type</a> , page 29. Default: <code>cascadeNone</code> .

**Table 103** `flex.participant.create` returned data

Parameter name	Type	Description
<code>participantID</code>	string (50)	Participant identifier assigned by the TelePresence Server. All subsequent invocations of commands to control or query this participant must use this identifier to reference it. See <a href="#">Identifiers and Client References, page 16</a> .
<code>participantReference</code>	string (50)	Client reference string. Returned if not empty. See <a href="#">Identifiers and Client References, page 16</a> .

`flex.participant.deletions.enumerate`

Enumerates only deleted participants.

The response will include either the `participantIDs` array or the `IDs` array, depending on the value of `extended` that you supply in the first invocation.

**Table 104** `flex.participant.deletions.enumerate` inputs

Parameter name	Type	Description
<code>cookie</code>	string (150)	Participant deletions enumeration cookie. This field must be absent at the start of the enumeration, and present (using the value returned by a previous invocation) to continue the enumeration. Default: none.
<code>max</code>	integer (> 0)	Maximum number of participant deletion records returned in response. If <code>max</code> is not specified, as many records are returned as is possible.
<code>conferenceID</code>	string (50)	Enumerates only participants in the specified conference. Can only be supplied when <code>cookie</code> is absent. Enumeration for non-existent conferences will fail. See <a href="#">Identifiers and Client References, page 16</a> . Default: none.
<code>extended</code>	boolean	If <code>true</code> , the response includes the <code>IDs</code> array. If <code>false</code> , the response includes the <code>participantIDs</code> array. <code>extended</code> is only accepted on the first enumerate command, and is ignored on subsequent enumerations. You cannot change the type of array returned during an enumeration.  Default: <code>false</code> .

**Table 105** `flex.participant.deletions.enumerate` returned data

Parameter name	Type	Description
<code>moreAvailable</code>	boolean	Whether there are more participant deletions to be enumerated.
<code>cookie</code>	string (150)	Cookie that must be supplied in the next invocation to continue the enumeration.
<code>participantIDs</code>	array of strings	Each member of the array is a string (50) that identifies a participant that has been deleted. See <a href="#">Identifiers and Client References, page 16</a> . Not returned if <code>IDs</code> is returned.
<code>IDs</code>	array of structs	Each member of the array is a struct that identifies a participant that has been deleted, and the conference from which that participant was deleted. See <a href="#">Table 106 IDs array struct members, page 87</a> .  Not returned if <code>participantIDs</code> is returned.

**Table 106** IDs array struct members

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
sipReasonHeader	string	<b>Optional.</b> If set, the first 63 characters of the disconnect Reason Header for SIP calls.

## flex.participant.destroy

Destroys the specified participant. Any existing calls are destroyed.

**Table 107** flex.participant.destroy inputs

Parameter name	Type	Description
participantID	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
allowExitScreen	boolean	<b>Optional.</b> If allowExitScreen is false then, the exitScreen setting for the conference is overridden and set false. Default is false.

## flex.participant.enumerate

Enumerates participants. The alternative command `flex.participant.advanced.enumerate` can be used instead, but you should only use one type of participant enumeration.

See [Enumeration, page 21](#) and [flex.participant.advanced.enumerate, page 80](#)

**Table 108** flex.participant.enumerate inputs

Parameter name	Type	Description
cookie	string (150)	Participant enumeration cookie. This field must be absent at the start of the enumeration, and present (using the value returned by a previous invocation) when continuing an enumeration. Default: none.
max	integer (> 0)	Maximum number of participant details to return in response. If max is not specified, as many records will be returned as is possible.
conferenceID	string (50)	Enumerates only participants in the specified conference. Can only be supplied when cookie is absent. Enumeration for non-existent conferences will fail. See <a href="#">Identifiers and Client References, page 16</a> . Default: none.

**Table 109** flex.participant.enumerate returned data

Parameter name	Type	Description
moreAvailable	boolean	Whether there are more participants to be enumerated.

**Table 109 flex.participant.enumerate returned data (continued)**

Parameter name	Type	Description
<code>cookie</code>	string (150)	Enumeration cookie that must be returned in the next invocation to continue the enumeration. See <a href="#">Enumeration, page 21</a> .
<code>participants</code>	array of structs	Each member of the array is a struct defining a participant. See <a href="#">Table 110 Participant information struct members, page 88</a> . This array may be empty.

**Table 110 Participant information struct members**

Parameter name	Type	Description
<code>participantID</code>	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>participantReference</code>	string (50)	Client reference string. See <a href="#">Identifiers and Client References, page 16</a> . Absent if empty.
<code>displayName</code>	string (80)	The current display name for this participant. Absent if empty.
<code>conferenceID</code>	string (50)	Conference to which the participant is connected. See <a href="#">Identifiers and Client References, page 16</a> .
<code>accessLevel</code>	string	Access level granted to the participant. See <a href="#">Table 7 Access level enumerated type, page 27</a> .  <b>Note:</b> Incoming calls are reported as <code>unknown</code> until they have been authorized (reach the lobby screen).
<code>connectionState</code>	string	One of <code>disconnected</code> , <code>connecting</code> , <code>connected</code> , or <code>onHold</code> . See <a href="#">Table 22 Participant connection state enumerated type, page 30</a> .
<code>calls</code>	array of structs	Each member of this array is a struct that defines one call associated with the participant. See <a href="#">Table 111 Participant call information struct members, page 88</a> .  If a call is absent, the array member is empty.  The array position of the call information struct matches the position of the corresponding address struct in the <code>addresses</code> array.
<code>addresses</code>	array of structs	Each member of this array is a struct that contains either the address or the URI for one of the calls associated with the participant.  See <a href="#">Table 112 Participant address struct members, page 89</a> .  The position in the array matches that of the associated call in the <code>calls</code> array.

**Table 111 Participant call information struct members**

Parameter name	Type	Description
<code>callID</code>	string (50)	Call identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>incoming</code>	boolean	Direction of the call: <code>true</code> indicates incoming; <code>false</code> indicates outgoing.



**Table 111 Participant call information struct members (continued)**

Parameter name	Type	Description
<code>address</code>	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.

**Table 112 Participant address struct members**

Parameter name	Type	Description
<code>URI</code>	string (80)	URI used or to be used by the endpoint to connect to the conference. Incoming calls only, not returned for outgoing calls.  Only returned if it is not empty.
<code>remoteAddress</code>	string (80)	Remote address specified in <a href="#">flex.participant.create, page 84</a> to call out to the endpoint. Outgoing calls only, not returned for incoming calls.  Only returned if it is not empty.

## flex.participant.media.enumerate

Enumerates participants for media information. A participant can consist of one or more calls.

The enumeration returns participants that have been newly added and calls that have had changes to token settings since the previous invocation of the method, as indicated by the cookie.

**Note:** Only one of the parameters `maxTokensPerChannelConfigured` and `maxTokensPerChannelConfiguredUnlimited` can be returned. See "[Unlimited](#)" Integers, [page 20](#).

**Table 113 flex.participant.media.enumerate inputs**

Parameter name	Type	Description
<code>cookie</code>	string (150)	Participant media enumeration cookie. This field must be absent at the start of the enumeration, and present (using the value returned by a previous invocation) when continuing an enumeration. Default: none.
<code>max</code>	integer (> 0)	Maximum number of participant media details to return in response. If <code>max</code> is not specified, as many records will be returned as is possible.
<code>conferenceID</code>	string (50)	Enumerates only participants in the specified conference. Can only be supplied when <code>cookie</code> is absent. Enumeration for non-existent conferences will fail. See <a href="#">Identifiers and Client References, page 16</a> . Default: none.

**Table 114 flex.participant.media.enumerate returned data**

Parameter name	Type	Description
<code>moreAvailable</code>	boolean	Whether there are more participants to be enumerated.
<code>cookie</code>	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
<code>participantMediaInfo</code>	array of structs	Each member of the array is a struct that defines the media token usage for the enumerated participant. The array may be empty if there is no data to return. See <a href="#">Table 115 Participant media information struct members, page 90</a> .

**Table 115 Participant media information struct members**

Parameter name	Type	Description
<code>participantID</code>	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>conferenceID</code>	string (50)	Conference identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>participantReference</code>	string (50)	Client reference string. See <a href="#">Identifiers and Client References, page 16</a> . Absent if empty.
<code>mainVideoTokenInfo</code>	struct	A struct that defines the tokens for main video. See <a href="#">Table 116 Participant token information struct members, page 90</a> .
<code>extendedVideoTokenInfo</code>	struct	A struct that defines the tokens for extended video (content). See <a href="#">Table 116 Participant token information struct members, page 90</a> .
<code>audioTokenInfo</code>	struct	A struct that defines the tokens for audio. See <a href="#">Table 116 Participant token information struct members, page 90</a> .
<code>creditsConfigured</code>	integer ( $\geq 0$ )	Number of credits configured for the participant.
<code>creditsFarEnd</code>	integer ( $\geq 0$ )	Number of credits required to match far end capability. Absent if the far-end capabilities are not yet known, or if some calls are not yet established.
<code>creditsNearEnd</code>	integer ( $\geq 0$ )	Number of credits required to match near end capability. Absent if the far-end capabilities are not yet known, or if some calls are not yet established.

**Table 116 Participant token information struct members**

Parameter name	Type	Description
<code>maxTokensConfigured</code>	integer ( $\geq 0$ )	Maximum number of tokens with respect to conference or participant configuration.
<code>maxTokensPerChannelConfigured</code>	integer ( $\geq 0$ )	Maximum number of tokens per channel with respect to the conference or participant configuration. Not returned if <code>maxTokensPerChannelConfiguredUnlimited</code> is returned ( <code>true</code> ).
<code>maxTokensPerChannelConfiguredUnlimited</code>	boolean	Whether there is an unlimited maximum number of tokens per channel with respect to the conference or participant configuration. Not returned if <code>maxTokensPerChannelConfigured</code> is returned, in which case it is implicitly <code>false</code> .
<code>maxTokensPerChannelFarEnd</code>	integer ( $\geq 0$ )	Maximum number of tokens per channel with respect to the capability of the far end. If the far end has a range of capabilities, these correspond to the maxima.  Absent if the far-end capabilities are not yet known, or if some calls are not yet established.

**Table 116 Participant token information struct members (continued)**

Parameter name	Type	Description
<code>maxTokensFarEnd</code>	integer ( $\geq 0$ )	Maximum number of tokens with respect to the capability of the far end. If the far end has a range of capabilities, these correspond to the maxima.  Absent if the far-end capabilities are not yet known, or if some calls are not yet established.
<code>maxTokensPerChannelNearEnd</code>	integer ( $\geq 0$ )	Maximum number of tokens per channel, advertised by the TelePresence Server.  Absent if the far-end capabilities are not yet known, or if some calls are not yet established.
<code>maxTokensNearEnd</code>	integer ( $\geq 0$ )	Maximum number of tokens advertised by the TelePresence Server.  Absent if the far-end capabilities are not yet known, or if some calls are not yet established.

## `flex.participant.modify`

Modifies the call attributes, media resources, and display name of the specified participant. Only the parameters that you specify are changed.

If you change the call attributes, your changes apply to all calls for this participant. Media resources are distributed as described in [Participant Media Distribution](#), page 19.

**Note:** The token requirements for a call cannot be known prior to instantiation of the call, so no checks are made on `flex.participant.create` or `flex.participant.modify` to determine if the call will have adequate resources. The client is therefore responsible for ensuring that the call has adequate resources.

**Table 117 `flex.participant.modify` inputs**

Parameter name	Type	Description
<code>participantID</code>	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References</a> , page 16.
<code>displayName</code>	string (80)	Configured display name for the endpoint. This overrides the endpoint display name setting.
<code>callAttributes</code>	struct	This <a href="#">Call Attributes Struct</a> , page 34 modifies the participant's call attributes. These settings override the conference default call attribute settings. See <a href="#">How Call Attributes are Derived</a> , page 34.
<code>participantMediaResources</code>	struct	This <a href="#">Participant Media Resources Struct</a> , page 33 modifies the participant's media resource configuration. These settings override the conference default participant media resource configuration.  If present, this struct updates all participant media configuration settings: unspecified optional fields are set to their default values.

## `flex.participant.query`

Returns the parameters of the specified participant.

**Table 118** flex.participant.query inputs

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .

**Table 119** flex.participant.query returned data

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
conferenceID	string (50)	Conference identifier. See <a href="#">Identifiers and Client References, page 16</a> .
PIN	string (40)	Numeric PIN. <b>Deprecated:</b> please use PINs instead
PINs	array of <a href="#">Participant PIN Definition, page 41</a>	List of PINs for this participant, if any of its calls are incoming. Maximum of 2.
callAttributes	struct	This is the previously specified or inherited <a href="#">Call Attributes Struct, page 34</a> of the queried participant. See <a href="#">How Call Attributes are Derived, page 34</a> . <b>Note:</b> The <code>accessLevel</code> for incoming calls is reported as <code>unknown</code> until they have been authorized (reached the lobby screen).
participantMediaResources	struct	This is the previously specified or inherited <a href="#">Participant Media Resources Struct, page 33</a> of the queried participant..
calls	array of structs	Each member of the array is a <a href="#">Participant Call Definition Struct, page 40</a> that contains the specifications of one of this participant's calls. The array will have at least one member and may have up to four.
camerasCrossed	boolean	Whether cameras in the group of endpoints specified by <code>calls</code> are crossed.
audioIndex	integer ( >= 0)	Position in the <code>calls</code> array of the call that will receive audio.
contentIndex	integer ( >= 0)	Position in the <code>calls</code> array of the call that will receive content.
cascadeRole	string	One of <code>cascadeNone</code> , <code>cascadeMaster</code> , or <code>cascadeSlave</code> . See <a href="#">Table 18 Cascade roles enumerated type, page 29</a> .
participantReference	string (50)	Client reference string. See <a href="#">Identifiers and Client References, page 16</a> . Returned if string length > 0.
displayName	string (80)	Configured display name for the endpoint. This overrides the endpoint display name setting. Returned if string length > 0.
dtmf	string (127)	Valid <a href="#">DTMF, page 23</a> characters in a sequence that is sent to the call nominated by the <code>audioIndex</code> . This sequence is only sent when dialing out and is not sent for incoming calls.

**Table 119** flex.participant.query returned data (continued)

Parameter name	Type	Description
callerName	string (80)	Calling name seen by the endpoint. Not used for incoming calls. Returned if string length > 0.
callerAddress	string (80)	Calling address seen by the endpoint. Not used for incoming calls. Returned if string length > 0.

## flex.participant.requestDiagnostics

Request call diagnostics for participants. If the participant has no active calls, the TelePresence Server returns fault code 56. See [Fault Codes](#), page 122.

**Table 120** flex.participant.requestDiagnostics inputs

Parameter name	Type	Description
participantID	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and client references</a> .
receiverURI	string (255)	<b>Required.</b> Fully-qualified http or https URI (for example, <a href="http://example.com:5050/RPC2">http://example.com:5050/RPC2</a> or <a href="https://example.com:5050/RPC2">https://example.com:5050/RPC2</a> ) to which the diagnostics are sent. If no port number is specified, the device uses the protocol defaults (80 and 443 respectively).
sourceIdentifier	string (255 ASCII)	Source identifier. If supplied, the identifier will be returned along with the participant diagnostics. If absent, the unit's Port A MAC address is given.

## Asynchronous reply

flex.participant.requestDiagnostics works asynchronously because the required information is not available immediately. Therefore, when the query is made for a particular participant (identified by participantID), a receiverURI needs to be provided. The diagnostics are sent back to the receiverURI. The message sent back is an XML-RPC methodCall with methodName participantDiagnosticsResponse and contains audioRx, audioTx, auxiliaryAudioRx, auxiliaryAudioTx, videoRx, videoTx, contentVideoRx, and contentVideoTx arrays, each of which contain a number of structs (one for each stream present). The member parameters of each struct type are described below.

The TelePresence Server can handle up to 10 concurrent asynchronous requests of this type, so this command may fail with fault code 203 if the number of pending requests exceeds this limit.

**Table 121** flex.participant.requestDiagnostics asynchronously returned data

Parameter name	Type	Description
participantID	string (50)	Identifier of the participant to which these diagnostics relate. See <a href="#">Identifiers and client references</a> .
sourceIdentifier	string	Source identifier provided in the original request. If absent, the unit's Ethernet A MAC address is given.
audioRx	array	Array of audioRx stream structs. See <a href="#">Table 122 audioRx stream struct members</a> , page 94.
audioTx	array	Array of audioTx stream structs. See <a href="#">Table 123 audioTx stream struct members</a> , page 94.

**Table 121 flex.participant.requestDiagnostics asynchronously returned data (continued)**

Parameter name	Type	Description
<code>auxiliaryAudioRx</code>	array	Array of <code>auxiliaryAudioRx</code> stream structs. See <a href="#">Table 124 auxiliaryAudioRx stream struct members, page 95</a> .
<code>auxiliaryAudioTx</code>	array	Array of <code>auxiliaryAudioTx</code> stream structs. See <a href="#">Table 125 auxiliaryAudioTx stream struct members, page 95</a> .
<code>videoRx</code>	array	Array of <code>videoRx</code> stream structs. See <a href="#">Table 126 videoRx stream struct members, page 96</a> .
<code>videoTx</code>	array	Array of <code>videoTx</code> stream structs. See <a href="#">Table 127 videoTx stream struct members, page 97</a> .
<code>contentVideoRx</code>	array	Array of <code>contentRx</code> stream structs. See <a href="#">Table 128 contentVideoRx stream struct members, page 97</a> .
<code>contentVideoTx</code>	array	Array of <code>contentTx</code> stream structs. See <a href="#">Table 129 contentVideoTx stream struct members, page 98</a> .

**Table 122 audioRx stream struct members**

Parameter name	Type	Description
<code>codec</code>	string	Codec in use.
<code>encrypted</code>	boolean	Whether the stream data is encrypted.
<code>channelBitRate</code>	integer	Bit rate of the channel in bits per second (bps).
<code>jitter</code>	integer	Current jitter in this stream, for transcoded streams: measured in milliseconds (ms).
<code>energy</code>	integer	Level of the signal, measured in decibels (dB).
<code>packetsReceived</code>	integer	Count of packets received in this stream.
<code>packetErrors</code>	integer	Count of packets with errors in this stream.
<code>packetsMissing</code>	integer	Count of packets missing from this stream.
<code>framesReceived</code>	integer	Count of frames received in this stream.
<code>frameErrors</code>	integer	Count of frames with errors in this stream.
<code>muted</code>	boolean	Whether the stream is muted.
<code>clearPathOverhead</code>	integer	Only returned if ClearPath has been negotiated. The percentage of FEC overhead in this media stream. The value 50, for example, means that one FEC packet is used to protect every two media packets.
<code>clearPathRecovered</code>	integer	Only returned if ClearPath has been negotiated. The number of media packets recovered using FEC.

**Table 123 audioTx stream struct members**

Parameter name	Type	Description
<code>codec</code>	string	Codec in use.

**Table 123 audioTx stream struct members (continued)**

Parameter name	Type	Description
<code>encrypted</code>	boolean	Whether the stream data is encrypted.
<code>channelBitRate</code>	integer	Bit rate of the channel in bits per second (bps).
<code>packetsSent</code>	integer	Count of packets sent in this stream.
<code>muted</code>	boolean	Whether the stream is muted.
<code>packetsLost</code>	integer	The number of packets lost from this stream, as reported by RTCP from the far end.
<code>clearPathOverhead</code>	integer	Only returned if ClearPath has been negotiated. The percentage of FEC overhead in this media stream. The value 50, for example, means that one FEC packet is used to protect every two media packets.
<code>clearPathRecovered</code>	integer	Only returned if ClearPath has been negotiated. The number of media packets recovered using FEC, as reported by RTCP from the far end.

**Table 124 auxiliaryAudioRx stream struct members**

Parameter name	Type	Description
<code>codec</code>	string	Codec in use.
<code>encrypted</code>	boolean	Whether the stream data is encrypted.
<code>channelBitRate</code>	integer	Bit rate of the channel in bits per second (bps).
<code>jitter</code>	integer	Current jitter in this stream, for transcoded streams: measured in milliseconds (ms).
<code>energy</code>	integer	Level of the signal, measured in decibels (dB).
<code>packetsReceived</code>	integer	Count of packets received in this stream.
<code>packetErrors</code>	integer	Count of packets with errors in this stream.
<code>packetsMissing</code>	integer	Count of packets missing from this stream.
<code>framesReceived</code>	integer	Count of frames received in this stream.
<code>frameErrors</code>	integer	Count of frames with errors in this stream.
<code>muted</code>	boolean	Whether the stream is muted.

**Table 125 auxiliaryAudioTx stream struct members**

Parameter name	Type	Description
<code>codec</code>	string	Codec in use.
<code>encrypted</code>	boolean	Whether the stream data is encrypted.
<code>channelBitRate</code>	integer	Bit rate of the channel in bits per second (bps).
<code>packetsSent</code>	integer	Count of packets sent in this stream.
<code>muted</code>	boolean	Whether the stream is muted.

**Table 126** videoRx stream struct members

Parameter name	Type	Description
streamType	string	One of: transcoded Or multistream. Indicates if the array element represents a transcoded or switched video stream. See notes below for switched streams.
codec	string	Codec in use.
height	integer	Height of the stream, in pixels. For switched streams: Provides the resolution requested by TS from the media source, although a lower resolution may be received.
width	integer	Width of the stream, in pixels. For switched streams: Provides the resolution requested by TS from the media source, although a lower resolution may be received.
encrypted	boolean	Whether the stream data is encrypted.
channelBitRate	integer	Bit rate of the channel in bits per second (bps).
expectedBitRate	integer	Expected bit rate of this stream, in bits per second (bps).
expectedBitRateReason	string	One of: viewedSize, errorPackets, or notLimited.
actualBitRate	integer	Measured bit rate of this stream, in bits per second (bps).
jitter	integer	Current jitter in this stream, for transcoded streams: measured in milliseconds (ms).
packetsReceived	integer	Count of packets received in this stream.
packetErrors	integer	Count of packets with errors in this stream.
framesReceived	integer	Count of frames received in this stream.
frameErrors	integer	Count of frames with errors in this stream.
frameRate	integer	Number of frames being received per second. For switched streams: Provides the framerate requested by TS from the media source, although a lower framerate may be received.
fastUpdateRequestsSent	integer	Number of fast update requests sent.
muted	boolean	Whether the stream is muted.
clearPathOverhead	integer	Only returned if ClearPath has been negotiated. The percentage of FEC overhead in this media stream. The value 50, for example, means that one FEC packet is used to protect every two media packets.
clearPathRecovered	integer	Only returned if ClearPath has been negotiated. The number of media packets recovered using FEC.
clearPathLTRFRepaired	integer	Only returned if ClearPath has been negotiated. The number of frames repaired by referencing the long-term reference frames embedded in this stream.



**Table 127** videoTx stream struct members

Parameter name	Type	Description
streamType	string	One of: <code>transcoded</code> or <code>multistream</code> . Indicates if the array element represents a transcoded or switched video stream. See notes below for switched streams.
codec	string	Codec in use.
height	integer	Height of the stream, in pixels. For switched streams: Provides the resolution requested by the media destination from TS, although a lower resolution may be sent.
width	integer	Width of the stream, in pixels. For switched streams: Provides the resolution requested by the media destination from TS, although a lower resolution may be sent.
encrypted	boolean	Whether the stream data is encrypted.
channelBitRate	integer	Bit rate of the channel in bits per second (bps).
configuredBitRate	integer	Configured bit rate of the channel (in bps), see <code>configuredBitRateReason</code> for why this differs from <code>channelBitRate</code> . The bitrate which TS requests from the media source.
configuredBitRateReason	string	One of: <code>aggregateBandwidth</code> , <code>flowControl</code> , or <code>notLimited</code> .
actualBitRate	integer	Measured bit rate of this stream, in bits per second (bps).
packetsSent	integer	Count of packets sent in this stream.
frameRate	integer	Number of frames being sent per second. For switched streams: Provides the framerate requested by the media destination from TS, although a lower framerate may be sent.
fastUpdateRequestsReceived	integer	Number of fast update requests received.
muted	boolean	Whether the stream is muted.
packetsLost	integer	The number of packets lost from this stream, as reported by RTCP from the far end.
clearPathOverhead	integer	Only returned if ClearPath has been negotiated. The percentage of FEC overhead in this media stream. The value 50, for example, means that one FEC packet is used to protect every two media packets.
clearPathRecovered	integer	Only returned if ClearPath has been negotiated. The number of media packets recovered using FEC, as reported by RTCP from the far end.
clearPathLTRF	boolean	Only returned if ClearPath has been negotiated. <code>true</code> if long-term reference frames are being inserted in this stream.

**Table 128** contentVideoRx stream struct members

Parameter name	Type	Description
streamType	string	Always <code>transcoded</code> .
codec	string	Codec in use.

**Table 128** contentVideoRx stream struct members (continued)

Parameter name	Type	Description
height	integer	Height of the stream, in pixels.
width	integer	Width of the stream, in pixels.
encrypted	boolean	Whether the stream data is encrypted.
channelBitRate	integer	Bit rate of the channel in bits per second (bps).
expectedBitRate	integer	Expected bit rate of this stream, in bits per second (bps).
expectedBitRateReason	string	One of: viewedSize, errorPackets, Or notLimited.
actualBitRate	integer	Measured bit rate of this stream, in bits per second (bps).
jitter	integer	Current jitter in this stream, for transcoded streams: measured in milliseconds (ms).
packetsReceived	integer	Count of packets received in this stream.
packetErrors	integer	Count of packets with errors in this stream.
framesReceived	integer	Count of frames received in this stream.
frameErrors	integer	Count of frames with errors in this stream.
frameRate	integer	Number of frames being received per second. For switched streams: Provides the framerate requested by TS from the media source, although a lower framerate may be received.
fastUpdateRequestsSent	integer	Number of fast update requests sent.
clearPathOverhead	integer	Only returned if ClearPath has been negotiated. The percentage of FEC overhead in this media stream. The value 50, for example, means that one FEC packet is used to protect every two media packets.
clearPathRecovered	integer	Only returned if ClearPath has been negotiated. The number of media packets recovered using FEC.
clearPathLTRFRepaired	integer	Only returned if ClearPath has been negotiated. The number of frames repaired by referencing the long-term reference frames embedded in this stream.

**Table 129** contentVideoTx stream struct members

Parameter name	Type	Description
streamType	string	Always transcoded.
codec	string	Codec in use.
height	integer	Height of the stream, in pixels.
width	integer	Width of the stream, in pixels.
encrypted	boolean	Whether the stream data is encrypted.
channelBitRate	integer	Bit rate of the channel in bits per second (bps).

**Table 129** contentVideoTx stream struct members (continued)

Parameter name	Type	Description
configuredBitRate	integer	Configured bit rate of the channel (in bps), see <code>configuredBitRateReason</code> for why this differs from <code>channelBitRate</code> .
configuredBitRateReason	string	One of: <code>aggregateBandwidth</code> , <code>flowControl</code> , Or <code>notLimited</code> .
actualBitRate	integer	Measured bit rate of this stream, in bits per second (bps).
packetsSent	integer	Count of packets sent in this stream.
frameRate	integer	Number of frames being sent per second. For switched streams: Provides the framerate requested by the media destination from TS, although a lower framerate may be sent.
fastUpdateRequestsReceived	integer	Number of fast update requests received.
packetsLost	integer	The number of packets lost from this stream, as reported by RTCP from the far end.
clearPathOverhead	integer	Only returned if ClearPath has been negotiated. The percentage of FEC overhead in this media stream. The value 50, for example, means that one FEC packet is used to protect every two media packets.
clearPathRecovered	integer	Only returned if ClearPath has been negotiated. The number of media packets recovered using FEC, as reported by RTCP from the far end.
clearPathLTRF	boolean	Only returned if ClearPath has been negotiated. <code>true</code> if long-term reference frames are being inserted in this stream.

## flex.participant.requestPreview

Requests JPEG previews of video streams to or from the specified participant.

`flex.participant.requestPreview` works asynchronously because participant previews are not available immediately. Therefore when the request is made for a particular participant (identified by `participantID`), a `receiverURI` needs to be provided.

The TelePresence Server can handle up to 10 concurrent asynchronous requests of this type, so this command may fail with fault code 203 if the number of pending requests exceeds this limit.

**Table 130** flex.participant.requestPreview inputs

Parameter name	Type	Description
participantID	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
receiverURI	string (255)	<b>Required.</b> Fully-qualified <code>http</code> or <code>https</code> URI (for example, <a href="http://example.com:5050/RPC2">http://example.com:5050/RPC2</a> or <a href="https://example.com:5050/RPC2">https://example.com:5050/RPC2</a> ) to which the previews are sent. If no port number is specified, the device uses the protocol defaults (80 and 443 respectively).
streams	array of structs	<b>Required.</b> Each member of the array is a <a href="#">Table 131 Stream struct inputs, page 100</a> that identifies which stream to preview. You must specify at least one stream and may specify up to four (if it is a grouped endpoint).

**Table 130 flex.participant.requestPreview inputs (continued)**

Parameter name	Type	Description
<code>sourceIdentifier</code>	string (255) ASCII characters only	Source identifier. If supplied, the identifier will be returned along with the previews. If absent, the MAC address of Port A is used.

**Table 131 Stream struct inputs**

Parameter name	Type	Description
<code>streamIdentifier</code>	string	<b>Required.</b> Video stream to preview. One of <code>rxMainVideo</code> , <code>txMainVideo</code> , or <code>extendedVideo</code> . In the case of <code>extendedVideo</code> , the choice of <code>incoming</code> or <code>outgoing</code> is decided by the TelePresence Server depending on what is currently active, and this is returned in the response.
<code>position</code>	integer ( $\geq 0$ and $\leq \{\text{number of screens} - 1\}$ )	This parameter must always be supplied unless the <code>streamIdentifier</code> is <code>extendedVideo</code> .  The parameter is only ever valid between 0 and 3, and defines the position of the endpoint stream within the group/multiscreen endpoint. This should always be 0 for a single-screen endpoint.  For grouped or multiscreen endpoints, 0 defines the leftmost screen and the number increments from left to right. For example, the right screen of a three-screen endpoint has <code>position=2</code> .
<code>maxWidth</code>	integer ( $\geq 88$ )	Maximum width of generated preview. Useful range 88-176 (pixels). Setting <code>maxWidth &gt; 176</code> will not return a wider image.  Default: 88.
<code>maxHeight</code>	integer ( $\geq 72$ )	Maximum height of generated preview. Useful range 72-144 (pixels). Setting <code>maxHeight &gt; 144</code> will not return a taller image.  Default: 72.

Examples of circumstances that cause this command to fail include the following:

- `streamIdentifier` is invalid (invalid parameter).
- `position` does not exist for the participant (invalid parameter).
- Values of `maxWidth` and `maxHeight` are invalid (invalid parameter).
- There are too many outstanding requests for previews (<Fault 203: 'too many asynchronous requests'>).
- `participantID` is invalid (no such participant).
- `receiverURI` is not a valid URI (invalid parameter).
- `streams` arrays is empty (invalid parameter).
- There is no active call in the slot indicated by `position`(<Fault 56: 'absent participant active call'>).

The maximum number of streams that are available to be requested is:

- $1 + (2 * \text{maximum\_number\_of\_calls\_per\_participant})$   
Where

- 1 stream is for extended video
- 2 streams per screen, incoming and outgoing.

For example, if the `maxCallsPerParticipant` returned by `flex.resource.query` is 4, a maximum of 9 streams are available to be requested.

## Asynchronous reply

If the request is successful, the previews of the requested streams are sent back to the `receiverURI`. The message sent back is an XML-RPC `methodCall` with `methodName` `participantPreviewResponse` and contains an array of `preview` structs - one for each `stream` supplied in the initial request.

**Table 132** `participantPreviewResponse` data

Parameter name	Type	Description
<code>participantID</code>	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>sourceIdentifier</code>	string (255)	Source identifier provided in the original request (or Port A MAC address if not supplied in request).
<code>streams</code>	array of structs	Each member of the array is a <a href="#">Table 133 Preview struct members, page 101</a> which will contain the base64-encoded JPEG preview if it was retrieved.

**Table 133** Preview struct members

Parameter name	Type	Description
<code>status</code>	string	Either <code>ok</code> , or one of the following strings giving the reason for failing to obtain a preview of the stream: <ul style="list-style-type: none"> <li>■ <code>audioOnly</code>: endpoint is audio-only and is not capable of receiving video.</li> <li>■ <code>noCurrentPresentation</code>: currently no active extended video channel (conference has no active presentation stream).</li> <li>■ <code>contentInMain</code>: there is active extended video, but this endpoint is not capable of receiving it - presentation is currently being displayed in <code>rxMainVideo</code> stream.</li> <li>■ <code>internalError</code>: unexpected error when trying to generate preview.</li> </ul> <p>The two content-related statuses are returned only if the requested stream is an extended video stream.</p>
<code>direction</code>	string	Whether the preview is of an <code>rx</code> (incoming) or <code>tx</code> (outgoing) stream.
<code>context</code>	string	Whether preview is of the <code>main</code> or <code>extended</code> stream.
<code>position</code>	integer	Position of stream starting from 0, going from left to right. For example, the right screen of a T3 would be <code>position = 2</code> .
<code>preview</code>	base64	Base64 encoded JPEG binary data (only valid if status is <code>ok</code> ). The image is constrained by <code>maxWidth</code> and <code>maxHeight</code> , and will be the size requested (up to 176x144), although the preview may not fill the returned image.

## `flex.participant.sendDTMF`

Sends the specified DTMF sequence to the endpoint nominated as the audio transmitter and receiver.

**Table 134** flex.participant.sendDTMF inputs

Parameter name	Type	Description
participantID	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
dtmf	string (127)	<b>Required.</b> Valid <a href="#">DTMF, page 23</a> characters in a sequence to send to this participant.

This command may fail with <Fault 56: 'absent participant active call'> if the call that has been nominated as the audio transmitter and receiver is absent.

## flex.participant.sendUserMessage

Sends the specified message to a particular participant. For multi-call participants, the message is sent to the call in the center.

The following table lists the input parameters that are required for this command.

**Table 135** flex.participant.sendUserMessage inputs

Parameter name	Type	Description
participantID	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
message	string (100)	<b>Required.</b> Message to display.
duration	integer (>0)	Duration, in seconds, for the message to display on the endpoint. The TelePresence Server will accept 0 but the behavior is undefined in this case. Default: 30.

## flex.participant.setImportant

Designates the specified participant as the important participant. This may result in importance being taken away from another participant in the same conference, even if the participant has no active calls.

**Table 136** flex.participant.setImportant inputs

Parameter name	Type	Description
participantID	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .

## flex.participant.setMute

Changes the muting states of the specified participant for incoming and outgoing audio and video streams. The muting state is only changed for fields that are specified in the command.

**Table 137** `flex.participant.setMute` inputs

Parameter name	Type	Description
<code>participantID</code>	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>audioRxMute</code>	boolean	Whether audio from the endpoint is muted.
<code>videoRxMute</code>	boolean	Whether the main video from the endpoint is muted.
<code>audioTxMute</code>	boolean	Whether audio to the endpoint is muted.
<code>videoTxMute</code>	boolean	Whether the main video to the endpoint is muted.

`flex.participant.status`

Returns the status of the specified participant.

**Table 138** `flex.participant.status` inputs

Parameter name	Type	Description
<code>participantID</code>	string (50)	<b>Required.</b> Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .

**Table 139** `flex.participant.status` returned data

Parameter name	Type	Description
<code>participantID</code>	string (50)	Participant identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>participantReference</code>	string (50)	Client reference string. See <a href="#">Identifiers and Client References, page 16</a> . Returned if string length > 0.
<code>displayName</code>	string (80)	Participant display name. Returned if string length > 0.
<code>conferenceID</code>	string (50)	Identifier of the conference to which the participant is connected or is in the process of connecting. See <a href="#">Identifiers and Client References, page 16</a> .
<code>accessLevel</code>	string	Access level assigned to the participant. See <a href="#">Table 7 Access level enumerated type, page 27</a> . <b>Note:</b> Incoming calls are reported as <code>unknown</code> until they have been authorized (reach the lobby screen).
<code>participantType</code>	string	Indicates the participant type. See <a href="#">Table 140 Participant type string values, page 104</a>
<code>participantMediaInfo</code>	struct	The <a href="#">Table 141 Participant media information struct members, page 105</a> details the media resources and credits allocated to this participant.
<code>important</code>	boolean	Whether the participant is important.

**Table 139 flex.participant.status returned data (continued)**

Parameter name	Type	Description
<code>calls</code>	array of structs	<p>Each member of the array is a struct that describes the status of one of the participant's calls. The array contains between one and four calls as previously configured for the participant.</p> <p>See <a href="#">Table 142 Participant call status struct members, page 105</a>.</p> <p>A member struct will be returned empty if the call is configured but does not currently exist. This ensures that the index position of each call is maintained, to assist you in determining the status of individual calls to grouped endpoints.</p>
<code>audioRxMute</code>	boolean	Whether audio from endpoint is muted.
<code>videoRxMute</code>	boolean	Whether main video from endpoint is muted.
<code>audioTxMute</code>	boolean	Whether audio to endpoint is muted.
<code>videoTxMute</code>	boolean	Whether main video to endpoint is muted.
<code>displayName</code>	string (80)	Participant display name. Returned if string length > 0.
<code>layout</code>	string	<p>The display layout that is currently shown on the participant's endpoint. One of <code>layoutSingle</code>, <code>layoutActivepresence</code>, <code>layoutProminent</code>, Or <code>layoutEqual</code>.</p> <p>Not returned if there are no calls connected for this participant or if no video is being transmitted to the participant (eg. if transmit video is muted).</p> <p>See <a href="#">Table 10 Single screen layout enumerated type, page 27</a> or <a href="#">Table 11 Multi-screen layout enumerated type, page 27</a>, depending on the type of endpoint.</p>

**Table 140 Participant type string values**

Value
Standard
Grouped endpoint
Group of 2 endpoints
Group of 3 endpoints
Group of 4 endpoints
Telepresence
TANDBERG T1
TANDBERG T3
TANDBERG T3 Custom Edition
TANDBERG TelePresence Server
SIP telepresence
SIP single screen telepresence



**Table 140 Participant type string values (continued)**

Value
SIP three screen telepresence
Legacy TIP endpoint
Legacy single screen TIP endpoint
Legacy three screen TIP endpoint
Cascade
Multistream
Unknown

**Table 141 Participant media information struct members**

Parameter name	Type	Description
<code>mainVideoTokenInfo</code>	struct	A Participant token information struct for the participant's main video. See <a href="#">Table 141 Participant media information struct members, page 105</a> .
<code>extendedVideoTokenInfo</code>	struct	A Participant token information struct for the participant's extended video. See <a href="#">Table 141 Participant media information struct members, page 105</a> .
<code>audioTokenInfo</code>	struct	A Participant token information struct for the participant's audio. See <a href="#">Table 141 Participant media information struct members, page 105</a> .
<code>creditsConfigured</code>	integer ( $\geq 0$ )	Number of credits configured for the participant.
<code>creditsFarEnd</code>	integer ( $\geq 0$ )	Number of credits required to match far end capability.  Absent if the far-end capabilities are not yet known, or if some calls are not yet established.
<code>creditsNearEnd</code>	integer ( $\geq 0$ )	Number of credits required to match near end capability.  Absent if the far-end capabilities are not yet known, or if some calls are not yet established.

**Table 142 Participant call status struct members**

Parameter name	Type	Description
<code>callID</code>	string (50)	Call identifier. See <a href="#">Identifiers and Client References, page 16</a> .
<code>conferenceID</code>	string (50)	Identifier of the conference to which the call is connected or is in the process of connecting. See <a href="#">Identifiers and Client References, page 16</a> .
<code>conferenceState</code>	string	State of call connection to the conference. See <a href="#">Table 16 Call conference state enumerated type, page 28</a> .
<code>callState</code>	string	State of the call. See <a href="#">Table 17 Call state enumerated type, page 28</a> .
<code>incoming</code>	boolean	Direction of the call: <code>true</code> indicates incoming; <code>false</code> indicates outgoing.

**Table 142 Participant call status struct members (continued)**

Parameter name	Type	Description
<code>protocol</code>	string	Call control protocol. See <a href="#">Table 12 Call protocol enumerated type, page 27</a> .
<code>address</code>	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.
<code>duration</code>	integer (>= 0)	Duration of the call in seconds. Only returned if a call has been established.
<code>rxBandwidth</code>	integer (>= 0)	Receive bandwidth. Only returned if a call has been established.
<code>txBandwidth</code>	integer (>= 0)	Transmit bandwidth. Only returned if a call has been established.
<code>remoteName</code>	string (80)	Endpoint name supplied by the far end. Only returned for strings of length > 0.

`flex.reservation.create` (experimental only—do not use)

**Table 143 `flex.reservation.create` inputs**

Parameter name	Type	Description
<code>reservationReference</code>	string (50)	Client reference. Default is blank.
<code>conferenceID</code>	string (50)	Conference identifier.
<code>participantRole</code>	string	What role the reservation is for.
<code>numParticipants</code>	integer	Number of participants the reservation is for.
<code>maxQuality</code>	integer	Maximum quality (in total credits) of each participant.

**Table 144 `flex.reservation.create` returned data**

Parameter name	Type	Description
<code>reservationId</code>	string (50)	Reservation identifier, assigned by the DSC. All subsequent queries for this reservation can use this identifier to reference the reservation. See <a href="#">Identifiers and Client References, page 16</a> .
<code>reservationReference</code>	string (50)	Client reference string, if not blank. See <a href="#">Identifiers and Client References, page 16</a> .

`flex.reservation.query` (experimental only—do not use)

**Table 145 `flex.reservation.query` inputs**

Parameter name	Type	Description
<code>reservationId</code>	string (50)	Reservation identifier, assigned by the DSC.

**Table 146** `flex.reservation.query` returned data

Parameter name	Type	Description
<code>reservationId</code>	string (50)	Reservation identifier, assigned by the DSC.
<code>reservationReference</code>	string (50)	Client reference.
<code>conferenceID</code>	string (50)	Conference identifier.
<code>participantRole</code>	string	What role the reservation is for.
<code>numParticipants</code>	integer	Number of participants the reservation is for.
<code>maxQuality</code>	integer	Maximum quality (in total credits) of each participant.
<code>totalReservedCredits</code>	integer	Total amount of credits reserved by the reservation.

`flex.reservation.modify` (experimental only—do not use)

**Table 147** `flex.reservation.modify` inputs

Parameter name	Type	Description
<code>reservationId</code>	string (50)	Reservation identifier, assigned by the DSC.
<code>reservationReference</code>	string (50)	Client reference.
<code>numParticipants</code>	integer	Number of participants the reservation is for.
<code>maxQuality</code>	integer	Maximum quality (in total credits) of each participant.

`flex.reservation.status` (experimental only—do not use)

**Table 148** `flex.reservation.status` inputs

Parameter name	Type	Description
<code>reservationID</code>	string (50)	<b>Required.</b> Reservation identifier, assigned by the DSC. See <a href="#">Identifiers and Client References, page 16</a> .

**Table 149** `flex.reservation.status` returned data

Parameter name	Type	Description
<code>reservationID</code>	string (50)	Reservation identifier assigned by the TelePresence Server. See <a href="#">Identifiers and Client References, page 16</a> .
<code>reservationReference</code>	string (50)	Client reference.
<code>numParticipantsRemaining</code>	integer	Number of participants still capable of using the reservation.
<code>numCreditsRemaining</code>	integer	Number of credits still reserved by the reservation.

## flex.reservation.enumerate (experimental only—do not use)

**Table 150** flex.reservation.enumerate inputs

Parameter name	Type	Description
cookie	string (150)	Reservation enumeration token. This field may be absent at the start of enumeration. Use the value returned by a previous call to continue the enumeration.
max	integer	Maximum number of reservation details to return in response.
conferenceID	string (50)	Filter, if present, only reservations associated with the conference the identifier refers to are returned.

**Table 151** flex.reservation.enumerate returned data

Parameter name	Type	Description
moreAvailable	boolean	1 if there are more participants to be enumerated, 0 if there are no more reservations to enumerate.
cookie	string (150)	Enumerate cookie which must be returned in the next invocation to continue the enumeration.
reservations	array	Array of reservation information structs. This array may be empty.

## flex.reservation.deletions.enumerate (experimental only—do not use)

**Table 152** flex.reservation.deletions.enumerate inputs

Parameter name	Type	Description
cookie	string (150)	Reservation deletions enumeration token. This field may be absent at the start of enumeration. Use the value returned by a previous call to continue the enumeration.
max	integer	Maximum number of reservation details to return in response.
conferenceID	string (50)	Filter, if present, only reservations associated with the conference the identifier refers to are returned.

**Table 153** flex.reservation.deletions.enumerate returned data

Parameter name	Type	Description
moreAvailable	boolean	1 if there are more participants to be enumerated, 0 if there are no more reservation deletions to enumerate.
cookie	string (150)	Enumerate cookie which must be returned in the next invocation to continue the enumeration.
IDs	array	The identifiers of reservations that have been deleted and their associated conference.

## flex.resource.query

Retrieves TelePresence Server resource settings/parameters. This command takes no input parameters.

## Deriving the required number of media tokens

You can derive the number of media tokens required to support a resolution by multiplying the required video width and height to get the required video area, and searching for the best fit in the `videoMediaTokenLevels` array. The best fit in this case is the lowest value of `maxVideoArea` that is larger than or equal to the required video area.

**Table 154** `flex.resource.query` returned data

Parameter name	Type	Description
<code>maxCalls</code>	integer ( $\geq 0$ )	Maximum number of active calls.
<code>maxCallsPerParticipant</code>	integer ( $\geq 0$ )	Maximum number of calls supported for any one participant.
<code>maxParticipants</code>	integer ( $\geq 0$ )	Maximum number of participants.
<code>maxParticipantsPerConference</code>	integer ( $\geq 0$ )	Maximum number of participants supported for any one conference.
<code>maxConferences</code>	integer ( $\geq 0$ )	Maximum number of conferences.
<code>maxMediaTokensPerChannel</code>	integer ( $> 0$ )	Maximum number of tokens that can be assigned to a channel.
<code>mediaTokensLimit</code>	integer ( $\geq 0$ )	Maximum number of media tokens available. This varies with the number of TelePresence Servers within the cluster.
<code>mediaTokensAvailable</code>	integer ( $\geq 0$ )	Number of media tokens currently available. This varies with the number of active TelePresence Servers within the cluster.
<code>maxMediaCredits</code>	integer ( $\geq 0$ )	Maximum number of credits available. This varies with installed screen licenses.
<code>mediaTokenLevelsMainVideo</code>	array of structs	Each member of the array is a <a href="#">Table 155 videoMediaTokenLevel struct members, page 110</a> struct that defines mediaToken levels for main video. The <code>videoMediaTokenLevel</code> XML-RPC struct is used to describe levels associated with mediaToken values such as the supported resolution.
<code>mediaTokenLevelsExtendedVideo</code>	array of structs	Each member of the array is a <a href="#">Table 155 videoMediaTokenLevel struct members, page 110</a> struct that defines mediaToken levels for extended video.
<code>mediaTokenLevelsAudio</code>	array of structs	Each member of the array is an <a href="#">Table 156 audioMediaTokenLevel struct members, page 110</a> that defines mediaToken levels for audio. The <code>audioMediaTokenLevel</code> struct is used to describe levels associated with mediaToken values required for audio channels.
<code>mediaCreditTokenRanges</code>	array of integers	Each entry is the top end (inclusive) of a media credit token range. The value of the previous entry + 1 is the bottom end of the range except for the very first range, which starts from 0. See <a href="#">Media Credits, page 20</a> .
<code>minCallBandwidth</code>	integer ( $> 0$ )	Lowest value (bits per second) that will be accepted for call bandwidths.
<code>maxCallBandwidth</code>	integer ( $> 0$ )	Highest value (bits per second) that will be accepted for call bandwidths.

**Table 155** videoMediaTokenLevel struct members

Parameter name	Type	Description
numMediaTokens	integer (>= 0)	Number of media tokens required for the given video resolution and macroblocks per second. See <a href="#">Deriving the required number of media tokens</a> .
maxVideoArea	integer (> 0)	Maximum resolution supported. Multiply required video width and height to check if a resolution is supported. A resolution is supported if: <code>maxVideoArea &gt;= (requiredVideoWidth * requiredVideoHeight)</code> .
maxMBps	integer (>= 0)	Maximum macroblocks per second.

**Table 156** audioMediaTokenLevel struct members

Parameter name	Type	Description
numMediaTokens	integer (>= 0)	Number of media tokens per channel.
stereo	boolean	Whether the channel is stereo.

## flex.resource.status

Returns resource usage information. This command takes no input parameters.

**Table 157** flex.resource.status returned data

Parameter name	Type	Description
numCalls	integer (>= 0)	Number of active calls.
numParticipants	integer (>= 0)	Number of active participants.
numConferences	integer (>= 0)	Number of active conferences.
configuredMediaTokens	integer (>= 0)	Total number of media tokens configured for use. That is, the sum of the tokens that have been configured for all participants in all conferences.
allocatedMediaTokens	integer (>= 0)	Total number of allocated media tokens. That is, the sum of tokens required for all participants in all conferences.
configuredMediaCredits	integer (>= 0)	Total number of license credits configured for use. That is, the sum of the credits that have been configured for all participants in all conferences.
allocatedMediaCredits	integer (>= 0)	Total number of allocated media credits. That is, the sum of credits required for all participants in all conferences.
conferenceCreditsAllocated (experimental only—do not use)	integer (>= 0)	Number of credits allocated to the conference (includes credits that are used and those that are reserved).

**Table 157 flex.resource.status returned data (continued)**

Parameter name	Type	Description
<code>conferenceCreditsRequired</code> (experimental only—do not use)	integer ( $\geq 0$ )	Number of credits needed by the conference to provide the highest quality experience to all participants. May be greater than <code>maxMediaCredits</code> .
<code>conferenceCreditsUsed</code> (experimental only—do not use)	integer ( $\geq 0$ )	Number of credits in use by this conference.

## logs.syslog.modify

Modifies the syslog settings of the device. Any fields not included are not modified.

**Table 158 logs.syslog.modify inputs**

Parameter name	Type	Description
<code>servers</code>	array	<b>Optional.</b> Structure for each syslog server to which the output should be sent. Maximum length 4. This array replaces all existing entries.
<code>address</code>	string (255)	<b>Optional.</b> Address of syslog server
<code>facilityValue</code>	integer	<b>Optional.</b> Facility value for use with all hosts, as defined by syslog protocol. Range 0 to 23.

## logs.syslog.query

Returns the current syslog settings of the device.

**Table 159 logs.syslog.query returned data**

Parameter name	Type	Description
<code>servers</code>	array	Structure for each syslog server to which we are sending our output. Maximum length 4.
<code>address</code>	string (255)	Address of syslog server
<code>packetsSent</code>	integer	Number of packets sent to server
<code>facilityValue</code>	integer	Facility value in use with all hosts, as defined by syslog protocol. Range 0 to 23.

## services.modify

Modifies the network services settings of the device. Any services not included in the array are left unmodified.

**Table 160** `services.modify` inputs

Parameter name		Type	Description
<code>ephemeralPortRangeMin</code>		integer	<b>Optional.</b> Number of the lowest port to be used for media traffic during conferencing. Can be as low as 10000. Must allow at least 5000 ports in this range.
<code>ephemeralPortRangeMax</code>		integer	<b>Optional.</b> Number of the highest port to be used for media traffic during conferencing. Can be as low as 15000. Must allow at least 5000 ports in this range.
<code>ports</code>		array	<b>Optional.</b> An array whose members are structures representing the ethernet port/protocol version combinations on the device.
	<code>ethernetPort</code>	string (10)	<b>Required for every entry in array.</b> Identifies which physical ethernet connector is being configured. One of <code>A</code> or <code>B</code> , though only <code>A</code> is available for use on current devices.
	<code>protocolVersion</code>	string (10)	<b>Required for every entry in array.</b> Identifies the IP protocol version, which may be <code>IPv4</code> or <code>IPv6</code> .
	<code>services</code>	array	<b>Required for every entry in array.</b> An array whose members are structures representing the services provided on the given port and protocol. Services not included in the array are not modified.
	<code>serviceName</code>	string (20)	<b>Required for every entry in array.</b> The name of the service. The available services may vary by hardware or software platform and version.
	<code>connectionType</code>	string (20)	<b>Required for every entry in array.</b> The type of service. Either <code>tcp</code> or <code>udp</code> .
	<code>setting</code>	boolean	<b>Optional for every entry in array.</b> <code>True</code> if the service should be enabled, <code>false</code> if it should be disabled.
	<code>portNumber</code>	integer	<b>Optional for every entry in array.</b> Identifies the port number on which the service is made available.

## services.query

Queries the network services settings of the device.

**Table 161** `services.query` returned data

Parameter name		Type	Description
Returned parameters			



**Table 161** services.query returned data (continued)

Parameter name		Type	Description
ports		array	An array whose members are structures representing the ethernet port/protocol combinations on the device.
	ethernetPort	string (10)	Identifies the Ethernet physical port. Always <sup>A</sup> on TelePresence Server.
	protocolVersion	string (10)	Identifies the IP protocol version. Either IPv4 or IPv6.
	services	array	An array whose members are structures representing the services provided on the given port and protocol. Services not included in the array are not modified.
	serviceName	string (20)	The name of the service. The available services may vary by hardware or software platform and version.
	connectionType	string (20)	The type of service. Either tcp or udp.
	setting	boolean	True if the service should be enabled, false if it should be disabled.
	portNumber	integer	Identifies the port number on which the service is made available.
	enabled	boolean	True if the service is actually enabled. Includes the setting, plus any requirement for feature key.
ephemeralPortRangeMin		integer	Number of the lowest port to be used for media traffic during conferencing.
ephemeralPortRangeMax		integer	Number of the highest port to be used for media traffic during conferencing.

## sip.modify

This call changes the TelePresence Server SIP settings.

**Table 162** sip.modify inputs

Parameter name	Type	Description
Optional parameters		
callConfiguration	string (20)	<b>Optional.</b> Either useTrunk or callDirect.
outboundAddress	string (80)	<b>Optional.</b> Address of SIP trunk destination. Ignored if callConfiguration is callDirect. May not be blank if callConfiguration is useTrunk.
outboundDomain	string (80)	<b>Optional.</b> The domain of the trunk destination. Ignored if callConfiguration is callDirect.
username	string (80)	<b>Optional.</b> The username used for SIP authentication.
password	string (63)	<b>Optional.</b> The password used for SIP authentication.

**Table 162 sip.modify inputs (continued)**

Parameter name	Type	Description
<code>outboundTransport</code>	string (3)	<b>Optional.</b> One of <code>udp</code> , <code>tcp</code> or <code>tls</code> . <code>tls</code> can only be configured when an encryption feature key is present.
<code>advertiseDualIPVersions</code>	string (20)	<b>Optional.</b> One of <code>disabled</code> or <code>useANAT</code> .
<code>negotiateSRTPUsingSDES</code>	string (20)	<b>Optional.</b> One of <code>TLS only</code> or <code>always</code> .
<code>useLocalCertificate</code>	boolean	<b>Optional.</b> True to use local certificate for connections and registrations. Ignored if not on 8710/7010 devices.

## sip.query

Returns the current TelePresence Server SIP settings.

**Table 163 sip.query returned data**

Parameter name	Type	Description
<code>callConfiguration</code>	string (20)	Either <code>useTrunk</code> or <code>callDirect</code> .
<code>outboundAddress</code>	string (80)	Address of SIP registrar or trunk destination. Ignored if <code>callConfiguration</code> is <code>callDirect</code> . May not be blank if <code>callConfiguration</code> is <code>useTrunk</code> .
<code>outboundDomain</code>	string (80)	The domain of the trunk destination. Ignored if <code>callConfiguration</code> is <code>callDirect</code> .
<code>username</code>	string (80)	The username used for SIP authentication.
<code>password</code>	string (63)	The password used for SIP authentication.
<code>outboundTransport</code>	string (3)	One of <code>udp</code> , <code>tcp</code> or <code>tls</code> .
<code>advertiseDualIPVersions</code>	string (20)	One of <code>disabled</code> or <code>useANAT</code> .
<code>negotiateSRTPUsingSDES</code>	string (20)	One of <code>TLS only</code> or <code>always</code> . Only returned when an encryption feature key is present.
<code>useLocalCertificate</code>	boolean	True to use local certificate for connections and registrations. Only returned on 8710/7010 devices.

## system.info

Returns the current status of the queried system. This command takes no input parameters.

**Table 164 system.info returned data**

Parameter name	Type	Description
<code>platform</code>	string	The TelePresence Server's platform, as it appears in <b>system.xml</b> .
<code>cpuModel</code>	string	The TelePresence Server's Hypervisor CPU model, as it appears in <b>system.xml</b> . (Cisco TelePresence Server on Virtual Machine only)
<code>cpuCount</code>	integer	The TelePresence Server's number of Virtual CPUs. (Cisco TelePresence Server on Virtual Machine only)

Table 164 system.info returned data (continued)

Parameter name	Type	Description
<code>operationMode</code>	string	One of <code>standalone</code> (locally managed), <code>flexible</code> (remotely managed), or <code>slave</code> (slave blade in a cluster).
<code>licenseMode</code>	string	Depends on the value of <code>operationMode</code> :  Either <code>HD</code> or <code>fullHD</code> , if <code>operationMode</code> is <code>standalone</code>  <code>flexible</code> if <code>operationMode</code> is <code>flexible</code> and in screen licensed mode  <code>trust</code> if <code>operationMode</code> is <code>flexible</code> and in Multiparty licensed mode  Absent if <code>operationMode</code> is <code>slave</code>
<code>numControlledServers</code>	integer	Number of TelePresence Servers controlled by this unit (including itself).
<code>clusterType</code>	string	The cluster status of this device. One of <code>master</code> , <code>slave</code> , or <code>unclustered</code> .
<code>depHash</code>	string	Build dependency hash. For development use.
<code>gateKeeperOK</code>	boolean	Whether the gatekeeper is configured and registered.
<code>tpsNumberOK</code>	integer	Number of configured and active TelePresence Servers.
<code>tpdVersion</code>	string	TelePresence Server version number.
<code>tpdName</code>	string	TelePresence Server system name.
<code>tpdUptime</code>	integer	Period of time (in seconds) that has passed since the system booted.
<code>tpdSerial</code>	string	TelePresence Server serial number.
<code>makeCallsOK</code>	boolean	In flexible (remotely managed) mode, this value is always <code>false</code> and should be ignored.
<code>portsVideoTotal</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>portsVideoFree</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>portsAudioTotal</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>portsAudioFree</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>portsContentTotal</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>portsContentFree</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>maxConferenceSizeVideo</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>maxConferenceSizeAudio</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.

**Table 164** `system.info` returned data (continued)

Parameter name	Type	Description
<code>maxConferenceSizeContent</code>	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.
<code>softwareVersion</code>	string	Software version string eg. <i>4.1</i>

## user.create

**Note:** `user` commands are only accessible by administrator users.

Users are identified by a `userid` which is a "username" string used to login to the device (with an associated password). The maximum number of users on the device is 200, after which no more can be created until some are destroyed.

There is an 'immortal' user (default id `admin`) with `administrator` access rights which always exists on the device. It cannot be destroyed, but can be modified by this API. Enumerations by administrators will include this 'immortal' user.

**Note:** We recommend not to overlap changes to the user list and enumerations.

The XML-RPC API client itself is authenticated using a user id and password. Three levels are defined:

<code>administrator</code>	This user can be used to authenticate API commands except for <code>user</code> commands.
<code>apiOnly</code>	This user can be used to authenticate API commands (in particular these ones). However API authenticated by such a user cannot modify or view <code>administrator</code> level users, including the <code>immortal</code> user.
<code>none</code>	This user cannot be used to authenticate any commands and is effectively disabled.

Creates a new user with the specified name and supplied parameters.

**Table 165** `user.create` inputs

Parameter name	Type	Description
<code>newUserId</code>	string (64)	<b>Required.</b> User ID with which the user will be able to login.
<code>password</code>	string (31)	<b>Required.</b> The password associated with the given user ID.
<code>accessRights</code>	string (32)	<b>Required.</b> One of <code>administrator</code> , <code>apiOnly</code> , <code>none</code> .
<code>name</code>	string (31)	<b>Optional.</b> Full name for the user. Blank if omitted.

## user.destroy

**Note:** `user` commands are only accessible by administrator users.

This call destroys a previously-created user.

**Table 166** `user.destroy` inputs

Parameter name	Type	Description
<code>currentUserId</code>	string (64)	<b>Required.</b> String with which the user can currently login. The <code>immortal</code> user cannot be destroyed with this API.

## user.modify

**Note:** `user` commands are only accessible by administrator users.

This call modifies the settings of a previously-created user. Only an administrator client can modify an administrator user. If a parameter is omitted, then the previous value of that attribute is not changed.

From version 4.2 or later, default administrator credentials must be changed on first login. No functionality or configuration is possible on the TelePresence Server until the default administrator credentials have been changed.

When using the API for first login, if the immortal administrator's password corresponds to the default password, then all API commands except: `user.modify` and `user.enumerate` will fail. Once the `user.modify` command has been used to change the immortal administrator's password, the TelePresence Server will function as usual.

**Table 167** `user.modify` inputs

Parameter name	Type	Description
<code>currentUserId</code>	string (64)	<b>Required.</b> User ID with which the user can currently login.
<code>newUserId</code>	string (64)	<b>Optional.</b> User ID with which the user will login, replacing the current user ID.
<code>name</code>	string (31)	<b>Optional.</b> Replacement full name for the user.
<code>password</code>	string (31)	<b>Optional.</b> Replacement password associated with the given user ID.
<code>accessRights</code>	string (32)	<b>Optional.</b> One of <code>administrator</code> , <code>apiOnly</code> , <code>none</code> .

## user.enumerate

**Note:** `user` commands are only accessible by administrator users.

This call enumerates all users configured on the system.

From version 4.2 or later, default administrator credentials must be changed on first login. No functionality or configuration is possible on the TelePresence Server until the default administrator credentials have been changed.

When using the API for first login, if the immortal administrator's password corresponds to the default password, then all API commands except: `user.modify` and `user.enumerate` will fail. Once the `user.modify` command has been used to change the immortal administrator's password, the TelePresence Server will function as usual.

**Note:** We recommend not to overlap changes to the user list and enumerations.

**Table 168** `user.enumerate` inputs

Parameter name	Type	Description
<code>enumerateID</code>	string (150)	<b>Optional.</b> User enumeration token. This field should be absent to start a fresh enumeration. Use the value returned by a previous call to continue the enumeration.
<code>max</code>	integer	<b>Optional.</b> Specifies maximum number of users to be returned by this call. The device may return fewer than 'max' users, if there are fewer remaining or if the response gets too large. If omitted, the client places no limit, and the number returned is determined purely by the device.

**Table 169** `user.enumerate` returned data

Parameter name	Type	Description
<code>moreAvailable</code>	boolean	Whether there are users remaining after this.
<code>enumerateID</code>	string (150)	The user enumeration token to be used to continue the enumeration. This will not be returned if the enumeration is complete.
<code>modifiedSinceStart</code>	boolean	Whether the user list may have been modified during this enumeration. This flag is for guidance only. Recommendation if this flag is set is to discard previous results and begin enumerating again.
<code>users</code>	array	List of the users, which are structures as follows:
<code>currentUserid</code>	string (64)	String with which the user can login.
<code>name</code>	string (31)	Full name for the user, or blank if none set.
<code>immortal</code>	boolean	<code>True</code> if this user is immortal (cannot be destroyed using this API).
<code>accessRights</code>	string (32)	Value of <code>administrator</code> , <code>apiOnly</code> or <code>none</code> .
<code>passwordChangeRequired</code>	boolean	<code>True</code> for users whose default credentials require changing on first login. This will not be returned once the password has been changed.

## Related Information

### `system.xml` on 8710 and 7010

You can derive some information about the TelePresence Server from its `system.xml` file. You can download this file via HTTP from the TelePresence Server's root.

#### Example `system.xml`

```
<?xml version="1.0"?>
<system>
  <manufacturer>TANDBERG</manufacturer>
  <model>Telepresence Server 8710</model>
  <product>TS</product>
  <platform>8710</platform>
  <productDisplayName>Cisco TelePresence Server</productDisplayName>
  <platformDisplayName>8710</platformDisplayName>
  <serial>SM021037</serial>
  <softwareVersion>4.1(1.22)</softwareVersion>
  <buildVersion>13.3(1.22)</buildVersion>
  <hostName>A host name</hostName>
  <ipAddress>198.51.100.14</ipAddress>
  <ipAddressV6>2001:DB8::81b7</ipAddressV6>
  <macAddress>BA:98:76:54:32:10</macAddress>
  <gatekeeperUsage>Yes</gatekeeperUsage>
  <gatekeeperAddress>mainvcs.test.lal</gatekeeperAddress>
  <gatekeeperIds>dt12b7,dt12b7-1,dt12b7-c,dt12b7-r</gatekeeperIds>
  <sipRegistrarUsage>Yes</sipRegistrarUsage>
  <sipRegistrarAddress>mainvcs.test.lal</sipRegistrarAddress>
</system>
```

```

<sipRegistrarDomain>test.lal</sipRegistrarDomain>
<sipTrunkUsage>No</sipTrunkUsage>
<sipTrunkAddress/>
<sipTrunkDomain/>
<isMaster>Yes</isMaster>
<clusterType>unclustered</clusterType>
<totalVideoPorts>12</totalVideoPorts>
<totalContentPorts>12</totalContentPorts>
<totalAudioOnlyPorts>10</totalAudioOnlyPorts>
<uptimeSeconds>230641</uptimeSeconds>
</system>

```

Table 170 System XML contents

Node name	Node contents
manufacturer	TANDBERG
model	Telepresence Server <model number> eg. <i>Telepresence Server 8710</i>
product	TS
platform	<platform> eg. <i>Media 310, 8710, or Virtual Machine with 16 vCPUs</i>
productDisplayName	<i>Cisco TelePresence Server</i> . The display name values are subject to change with new software releases, so your application should not rely on them.
platformDisplayName	<platform> eg. <i>Media 310, 8710, or Virtual Machine with 16 vCPUs</i> . The display name values are subject to change with new software releases, so your application should not rely on them.
serial	Unique serial number of the unit
softwareVersion	Software version string eg. <i>4.1(1.22)</i>
buildVersion	Build number string eg. <i>13.3(1.22)</i>
hostName	Host name of the unit
ipAddress	IPv4 address
ipAddressV6	IPv6 address
macAddress	MAC address
gatekeeperUsage	<b>yes</b> : gatekeeper usage is enabled <b>no</b> : gatekeeper usage is disabled
gatekeeperAddress	The gatekeeper host name or IP address
gatekeeperIds	Comma separated list of registered IDs associated with this TelePresence Server and its slaves (omitted if the system is not a master)
sipRegistrarUsage	<b>yes</b> : registrar usage is enabled <b>no</b> : registrar usage is disabled  This value is always <b>no</b> in remotely managed mode.
sipRegistrarAddress	SIP registrar host name / IP address  This node is always empty in remotely managed mode.

**Table 170 System XML contents (continued)**

Node name	Node contents
sipRegistrarDomain	SIP registrar domain  This node is always empty in remotely managed mode.
sipTrunkUsage	<b>Yes:</b> trunk usage is enabled <b>No:</b> trunk usage is disabled
sipTrunkAddress	SIP trunk host name / IP address
sipTrunkDomain	SIP trunk domain
isMaster	<b>Yes:</b> this system is a master, or it is unclustered <b>No:</b> this system is a slave
clusterType	The role of this system in a cluster. May be <code>unclustered</code> , <code>master</code> , or <code>slave</code>
totalVideoPorts	Total number of video ports  This value is always 0 and should be ignored.
totalContentPorts	Total number of video content ports  This value is always 0 and should be ignored.
totalAudioOnlyPorts	Total number of audio-only ports  This value is always 0 and should be ignored.
uptimeSeconds	System uptime in seconds

## system.xml on Media 310/320

You can derive some information about the TelePresence Server from its **system.xml** file. You can download this file via HTTP from the TelePresence Server's root.

### Example system.xml

```
<?xml version="1.0"?>
  <system>
    <manufacturer>Cisco</manufacturer>
    <model>TelePresence Server on Media 320</model>
    <product>TS</product>
    <platform>Media 320</platform>
    <productDisplayName>Cisco TelePresence Server</productDisplayName>
    <platformDisplayName>Media 320</platformDisplayName>
    <serial>SUK1702000D</serial>
    <softwareVersion>4.1(1.22)</softwareVersion>
    <buildVersion>13.3(1.22)</buildVersion>
    <hostName>HostName1</hostName>
    <ipAddress>198.51.100.15</ipAddress>
    <ipAddressV6>2001:DB8::81b8</ipAddressV6>
    <macAddress>01:23:45:67:89:AB</macAddress>
    <clusterType>unclustered</clusterType>
  </system>
```



**Table 171 System XML contents**

Node name	Node contents
manufacturer	Cisco
model	TelePresence Server on <platform> eg. <i>TelePresence Server on Media 310</i>
product	TS
platform	<platform> eg. <i>Media 310, 8710, or Virtual Machine with 16 vCPUs</i>
productDisplayName	<i>Cisco TelePresence Server</i> . The display name values are subject to change with new software releases, so your application should not rely on them.
platformDisplayName	<platform> eg. <i>Media 310, 8710, or Virtual Machine with 16 vCPUs</i> . The display name values are subject to change with new software releases, so your application should not rely on them.
serial	Unique serial number of the unit
softwareVersion	Software version string eg. <i>4.1(1.22)</i>
buildVersion	Build number string eg. <i>13.3(1.22)</i>
hostName	Host name of the unit
ipAddress	IPv4 address
ipAddressV6	IPv6 address
macAddress	MAC address
clusterType	The role of this system in a cluster. May be <i>unclustered, master, or slave</i>

## system.xml on Virtual Machine

You can derive some information about the TelePresence Server from its **system.xml** file. You can download this file via HTTP from the TelePresence Server's root.

### Example system.xml

```
<?xml version="1.0"?>
<system>
  <manufacturer>Cisco</manufacturer>
  <model>TelePresence Server on Virtual Machine with 16 vCPUs</model>
  <cpuModel>Intel (R) Xeon (R) CPU E5-4650 0 @ 2.70GHz</cpuModel>
  <cpuCount>30</cpuCount>
  <cpuAvx>1</cpuAvx>
  <product>TS</product>
  <platform>Virtual Machine with 16 vCPUs</platform>
  <productDisplayName>Cisco TelePresence Server</productDisplayName>
  <platformDisplayName>Virtual Machine with 16 vCPUs</platformDisplayName>
  <serial>057ED0A9</serial>
  <softwareVersion>4.1(1.22)</softwareVersion>
  <buildVersion>13.3(1.22)</buildVersion>
  <hostName>HostName1</hostName>
  <ipAddress>198.51.100.15</ipAddress>
  <ipAddressV6>2001:DB8::81b8</ipAddressV6>
  <macAddress>01:23:45:67:89:AB</macAddress>
  <clusterType>unclustered</clusterType>
</system>
```

**Table 172 System XML contents**

Node name	Node contents
manufacturer	Cisco
model	TelePresence Server on <platform> eg. <i>TelePresence Server on Virtual Machine with 16 vCPUs</i>
cpuModel	TelePresence Server's Hypervisor CPU model (e.g. Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz, etc.) (Cisco TelePresence Server on Virtual Machine only)
cpuCount	TelePresence Server's number of virtual CPUs (Cisco TelePresence Server on Virtual Machine only)
cpuAvx	TelePresence Server's support for AVX instruction. <i>0</i> if AVX is not supported or <i>1</i> if AVX is supported. (Cisco TelePresence Server on Virtual Machine only)
product	TS
platform	<platform> eg. <i>Media 310, 8710, or Virtual Machine with 16 vCPUs</i>
productDisplayName	<i>Cisco TelePresence Server</i> . The display name values are subject to change with new software releases, so your application should not rely on them.
platformDisplayName	<platform> eg. <i>Media 310, 8710, or Virtual Machine with 16 vCPUs</i> . The display name values are subject to change with new software releases, so your application should not rely on them.
softwareVersion	Software version string eg. <i>4.1(1.22)</i>
buildVersion	Build number string eg. <i>13.3(1.22)</i>
hostName	Host name of the unit
ipAddress	IPv4 address
ipAddressV6	IPv6 address
macAddress	MAC address
clusterType	Always <b>unclustered</b> . (Cisco TelePresence Server on Virtual Machine does not support clustering.)

## Fault Codes

The Cisco TelePresence Server returns a fault code when it encounters a problem with processing an XML-RPC request.

The following table lists the fault codes that may be returned by the TelePresence Server and their most common interpretations.

**Table 173 Fault codes**

Fault Code	Description
1	<b>method not supported</b> . This method is not supported on this device or is unknown.
4	<b>no such conference</b> . The conference identification given does not match any conference.
5	<b>no such participant</b> . The participant identification given does not match any participants.

**Table 173 Fault codes (continued)**

6	<code>too many conferences</code> . The device has reached the limit of the number of conferences that can be configured.
7	<code>too many participants</code> . There are already too many participants configured and no more can be created.
14	<code>authorization failed</code> . The requested operation is not permitted because the supplied authentication parameters were not recognized.
15	<code>insufficient privileges</code> . The specified user id and password combination is not valid for the attempted operation.
17	<code>call reservation failure</code> . There are insufficient free calls/participants to complete/place the requested calls.
18	<code>duplicate URI</code> . A URI was given, but this URI is already in use.
20	<code>unsupported participant type</code> . A participant type was used which does not correspond to any participant type known to the device.
25	<code>participant limit lower than active</code> . New participant limit is lower than current number of participants.
33	<code>out of range</code> . A call supplied a value that is outside of the allowed range for this parameter.
34	<code>internal error</code> . An error occurred while processing the API request.
35	<code>string is too long</code> . The call supplied a string parameter that was longer than allowed.
42	<code>port conflict</code> . The call attempts to set a port number that is already in use by another service.
49	<code>operation would disable active interface</code> .
50	<code>binary data array is too long</code> . The call supplied binary data that was longer than allowed.
52	<code>no available SIP registration</code> . There is no available SIP registration to complete the call.
53	<code>insufficient media credits or tokens</code> . Fewer media credits or tokens were supplied than were required to complete the call.
55	<code>malformed cookie</code> . The supplied cookie could not be read.
56	<code>no active participant call</code> . The participant does not currently have any active calls, or has no active call at the specified position.
57	<code>some participants failed</code> . The API request could not be completed for some participants in a conference.
58	<code>incorrect media credits or tokens</code> . Fewer media credits or tokens were supplied than are currently in use.
60	<code>IP setting is invalid</code> . IP setting is invalid
61	The removal of a feature or license key failed for one of several reasons. The fault code message will vary depending on the underlying cause of the failure.
63	<code>Attempt to modify immortal user</code> . Impossible to change the access rights or delete the immortal user.
64	<code>Attempt to delete unknown user</code> . It was not possible to delete the user with the given ID, because there is no such user.
65	<code>Invalid trust secret</code> . An invalid trust secret has been specified.
66	<code>Invalid key</code> . Key not recognised or has expired.
67	<code>Active conferences</code> . There are currently one or more conferences on this TS.

**Table 173 Fault codes (continued)**

101	<code>missing parameter</code> . This is given when a required parameter is absent. The parameter in question is given in the fault string in the format "missing parameter: parameter_name".
102	<code>invalid parameter</code> . This is given when a parameter was successfully parsed, is of the correct type, but falls outside the valid values; for example an integer is too high or a string value for an enumerated type contains an invalid value.
103	<code>malformed parameter</code> . This is given when a parameter of the correct name is present, but cannot be read for some reason; for example the parameter is supposed to be an integer, but is given as a string. The parameter in question is given in the fault string in the format "malformed parameter: parameter_name".
105	<code>request too large</code> . The method call contains more data than the API can accept. The maximum size of the call is 32 kilobytes.
201	<code>operation failed</code> . This is a generic fault for when an operation does not succeed as required.
202	<code>Product needs its activation feature key</code> . This request requires that the product is activated.
203	<code>Too many asynchronous requests</code> . The TelePresence Server is currently dealing with the maximum number of asynchronous requests of this type. Please retry this request later.
204	<code>Too many invalid keys entered. Wait 5 seconds to retry</code> . The TelePresence Server will not currently accept more requests to add feature keys.
205	<code>Duplicate user id</code> . A user id was given, but there is already a user with the same id.
206	<code>HTTPS required</code> . Certain requests are required to be made over HTTPS when the TelePresence Server is operating in trust licensed mode.
207	<code>System has been shut down</code> . No conferences or participants are created when the system has been shut down

Example XML-RPC Response to `flex.conference.create`

## Method call

```

<?xml version='1.0' encoding='UTF-8'?>
<methodCall>
  <methodName>flex.conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationPassword</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>Flex API conference</string>
            </value>
          </member>
          <member>
            <name>participantMediaResources</name>
            <value>
              <struct>
                <member>
                  <name>mediaTokensAudio</name>
                  <value>
                    <struct>
                      <member>
                        <name>total</name>
                        <value>
                          <int>96</int>
                        </value>
                      </member>
                    </struct>
                  </value>
                </member>
                <member>
                  <name>mediaTokensExtendedVideo</name>
                  <value>
                    <struct>
                      <member>
                        <name>total</name>
                        <value>
                          <int>1920</int>
                        </value>
                      </member>
                    </struct>
                  </value>
                </member>
                <member>
                  <name>mediaTokensMainVideo</name>
                  <value>
                    <struct>
                      <member>
                        <name>total</name>
                        <value>
                          <int>1920</int>
                        </value>
                      </member>
                    </struct>
                  </value>
                </member>
              </struct>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>

```

```

        </value>
      </member>
    <member>
      <name>numMediaCredits</name>
      <value>
        <int>5040</int>
      </value>
    </member>
  </struct>
</value>
</member>
<member>
  <name>authenticationUser</name>
  <value>
    <string>admin</string>
  </value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>

```

## Method response

```

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>conferenceID</name>
            <value>
              <string>b9852090-f5b9-11e1-8ac5-000d071080b8</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>

```

## Remotely Managed API Change History

Table 174 API version 4.3 change summary

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.conference.create, page 57</a>	customPINEntryFailedMessage useCustomPINEntryFailedMessage customDisconnectPlatitudeMessage useCustomDisconnectPlatitudeMessage customWelcomeScreenAudio useCustomWelcomeScreenAudio customPINEntryAudio useCustomPINEntryAudio customOptionalPINEntryAudio useCustomOptionalPINEntryAudio customPINIncorrectAudio useCustomPINIncorrectAudio customConferenceEndedExitAudio useCustomConferenceEndedExitAudio customParticipantDisconnectedExitAudio useCustomParticipantDisconnectedExitAudio customPINFailedExitAudio useCustomPINFailedExitAudio customWaitingForChairAudio useCustomWaitingForChairAudio customOnlyParticipantAudio useCustomOnlyParticipantAudio resourceOptimizationMode customBackgroundImageUrl	New parameters
	useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage	Removed
<a href="#">flex.conference.modify, page 67</a>	customPINEntryFailedMessage useCustomPINEntryFailedMessage customDisconnectPlatitudeMessage useCustomDisconnectPlatitudeMessage	New parameters
	useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage	Removed

Table 174 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.conference.query, page 72</a>	customPINEntryFailedMessage useCustomPINEntryFailedMessage customDisconnectPlatitudeMessage useCustomDisconnectPlatitudeMessage customWelcomeScreenAudio useCustomWelcomeScreenAudio customPINEntryAudio useCustomPINEntryAudio customOptionalPINEntryAudio useCustomOptionalPINEntryAudio customPINIncorrectAudio useCustomPINIncorrectAudio customConferenceEndedExitAudio useCustomConferenceEndedExitAudio customParticipantDisconnectedExitAudio useCustomParticipantDisconnectedExitAudio customPINFailedExitAudio useCustomPINFailedExitAudio customWaitingForChairAudio useCustomWaitingForChairAudio customOnlyParticipantAudio useCustomOnlyParticipantAudio resourceOptimizationMode customBackgroundImageURL	New parameters
	useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage	Removed
<a href="#">flex.conference.sendUserMessage, page 78</a>	message	Limited to 100 text characters
<a href="#">flex.conference.sendUserMessage, page 78</a>	position	Removed
<a href="#">flex.participant.sendUserMessage, page 102</a>	message	Limited to 100 text characters
<a href="#">flex.participant.sendUserMessage, page 102</a>	position	Removed
<a href="#">flex.conference.status, page 78</a>	conferenceCreditsAllocated conferenceCreditsRequired conferenceCreditsUsed	New parameters (experimental only—do not use)
<a href="#">flex.resource.status, page 110</a>	conferenceCreditsAllocated conferenceCreditsRequired conferenceCreditsUsed	New parameters (experimental only—do not use)



Table 174 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.conference.enumerate</a> , page 64	conferenceCreditsAllocated conferenceCreditsRequired conferenceCreditsUsed	New parameters (experimental only—do not use)
Table 28 Participant role enumerated type (experimental only—do not use), page 32	any accessLevelChair accessLevelGuest cascade	New enumerated type (experimental only—do not use)
Table 29 Resource optimization mode enumerated type, page 32	byParticipant byConference (experimental only—do not use)	New enumerated type
Enumerated Reservation Information Struct (experimental only—do not use), page 41	reservationID reservationReference conferenceID participantRole numParticipants maxQuality totalReservedCredits numParticipantsRemaining numCreditsRemaining	New struct (experimental only—do not use)
<a href="#">flex.reservation.create</a> (experimental only—do not use), page 106	reservationReference conferenceID participantRole numParticipants maxQuality reservationId reservationReference	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.query</a> (experimental only—do not use), page 106	reservationId reservationReference conferenceID participantRole numParticipants maxQuality totalReservedCredits	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.modify</a> (experimental only—do not use), page 107	reservationId reservationReference numParticipants maxQuality	New command and parameters (experimental only—do not use)

Table 174 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.reservation.status (experimental only—do not use)</a> , page 107	reservationID reservationReference numParticipantsRemaining numCreditsRemaining	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.destroy (experimental only—do not use)</a> , page 1	reservationId	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.enumerate (experimental only—do not use)</a> , page 108	cookie max conferenceID moreAvailable reservations	New command and parameters (experimental only—do not use)
<a href="#">flex.reservation.deletions.enumerate (experimental only—do not use)</a> , page 108	cookie max conferenceID moreAvailable IDs	New command and parameters (experimental only—do not use)

Table 175 API version 4.2 change summary

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.licenseMode.modify</a> , page 79	newLicenseMode, version, trustSecret	New command
<a href="#">flex.licenseMode.verify</a> , page 80	None	New command
<a href="#">system.info</a> , page 114	licenseMode; new options flexible, trust	Updated parameter
<a href="#">user.enumerate</a> , page 117	passwordChangeRequired	New parameter
<a href="#">Table 32 callAttributes struct members</a> , page 35	displayShowEndpointNames	Corrected default to True

**Table 176 API version 4.1.2 change summary**

XML-RPC Request / Topic	Parameter	Change
<a href="#">device.network.modify, page 47</a>	portA, dns, ipv4Enabled, dhcpv4, ipv4Address, ipv4SubnetMask, defaultIpv4Gateway, ipv6Address, ipv6Conf, ipv6PrefixLength, defaultIpv6Gateway, ethernetAutomatic, speed, fullDuplex, dnsConfiguration, hostName, nameServer, nameServerSecondary, domainName	New command
<a href="#">device.time.modify, page 52</a>	currentTime, ntpEnabled, utcOffsetHours, utcOffsetMinutes, ntpHost	New command
<a href="#">device.time.query, page 53</a>	currentTime, ntpEnabled, utcOffsetHours, utcOffsetMinutes, ntpHost	New command
<a href="#">logs.syslog.modify, page 111</a>	servers, facilityValue	New command
<a href="#">logs.syslog.query, page 111</a>	servers, facilityValue	New command
<a href="#">services.modify, page 111</a>	ephemeralPortRangeMin, ephemeralPortRangeMax, ports, ethernetPort, protocolVersion, services, serviceName, connectionType, setting, portNumber	New command
<a href="#">services.query, page 112</a>	ports, ethernetPort, protocolVersion, services, serviceName, connectionType, setting, portNumber, enabled, ephemeralPortRangeMin, ephemeralPortRangeMax	New command
<a href="#">sip.modify, page 113</a>	callConfiguration, outboundAddress, outboundDomain, username, password, outboundTransport, advertiseDualIPVersions, negotiateSRTPUsingSDES, useLocalCertificate	New command
<a href="#">sip.query, page 114</a>	callConfiguration, outboundAddress, outboundDomain, username, password, outboundTransport, advertiseDualIPVersions, negotiateSRTPUsingSDES, useLocalCertificate	New command
<a href="#">user.create, page 116</a>	newUserId, password, accessRights, name	New command
<a href="#">user.destroy, page 116</a>	currentUserId	New command
<a href="#">user.enumerate, page 117</a>	enumerateID, max, moreAvailable, modifiedSinceStart, users, currentUserId, name, immortal, accessRights	New command

**Table 176 API version 4.1.2 change summary (continued)**

XML-RPC Request / Topic	Parameter	Change
<a href="#">user.modify</a> , page 117	<code>currentUserId</code> , <code>newUserId</code> , <code>name</code> , <code>password</code> , <code>accessRights</code>	New command

**Table 177 API version 4.1 change summary**

XML-RPC Request / Topic	Parameter	Change
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>packetLossThreshold</code>	Modified default from 0 to 5
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>accessLevel</code>	New value <code>unknown</code> added
<a href="#">flex.participant.query</a> , page 91	<code>callAttributes</code>	Add note regarding <code>unknown</code> status
<a href="#">flex.participant.enumerate</a> , page 87	<code>accessLevel</code>	Add note regarding <code>unknown</code> status
<a href="#">flex.participant.advanced.enumerate</a> , page 80	<code>accessLevel</code>	Add note regarding <code>unknown</code> status
<a href="#">flex.participant.status</a> , page 103	<code>accessLevel</code>	Add note regarding <code>unknown</code> status
<a href="#">Table 16 Call conference state enumerated type</a> , page 28	<code>disconnecting</code>	Added
<a href="#">Table 17 Call state enumerated type</a> , page 28	<code>callStateFailed</code>	New status added
<a href="#">Table 21 Multistream mode enumerated type</a> , page 30		New function and parameters
<a href="#">Table 22 Participant connection state enumerated type</a> , page 30	<code>disconnecting</code> <code>neverConnected</code> <code>deferred</code> <code>partiallyFailed</code> <code>audioReceiverFailed</code>	Add new parameters to Participant connection state enumerated type table
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>indicateAudioOnlyParticipants</code>	Added
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>displayForceDefaultLayout</code>	Added Note
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>iXEnabled</code>	Modified default to True
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>displayLayoutSwitchingMode</code>	Added Note
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>multistreamMode</code>	Added
<a href="#">Table 32 callAttributes struct members</a> , page 35	<code>displayHighlightActiveSpeaker</code>	Parameter removed
<a href="#">Table 35 Outgoing participant call definition struct members</a> , page 40	<code>toOverride</code>	New Outgoing parameter added

Table 177 API version 4.1 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
<a href="#">Participant PIN Definition, page 41</a>		New set of parameters for incoming calls
<a href="#">flex.conference.create, page 57</a>	useCustomOptionalPINEntryMessage customOptionalPINEntryMessage exitScreen useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage useCustomConferenceEndedExitMessage customConferenceEndedExitMessage useCustomParticipantDisconnectedExitMessage customParticipantDisconnectedExitMessage	Added
<a href="#">flex.conference.modify, page 67</a>	useCustomOptionalPINEntryMessage customOptionalPINEntryMessage exitScreen useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage useCustomConferenceEndedExitMessage customConferenceEndedExitMessage useCustomParticipantDisconnectedExitMessage customParticipantDisconnectedExitMessage	Added
<a href="#">flex.conference.destroy, page 64</a>	allowExitScreen	Added
<a href="#">flex.conference.query, page 72</a>	useCustomOptionalPINEntryMessage customOptionalPINEntryMessage exitScreen useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage useCustomConferenceEndedExitMessage customConferenceEndedExitMessage useCustomParticipantDisconnectedExitMessage customParticipantDisconnectedExitMessage	Added

**Table 177 API version 4.1 change summary (continued)**

XML-RPC Request / Topic	Parameter	Change
Table 79 Conference information struct, page 66	<code>terminating</code>	Added
<code>flex.conference.sendUserMessage</code> , page 78	<code>position</code>	Modified default from 5 to 2
<code>flex.participant.create</code> , page 84	PIN	Deprecated
<code>flex.participant.create</code> , page 84	PINs	Added
<code>flex.participant.query</code> , page 91	PIN	Deprecated
<code>flex.participant.query</code> , page 91	PINs	Added
Table 140 Participant type string values, page 104		Add new table of string values
<code>flex.participant.destroy</code> , page 87	<code>allowExitScreen</code>	Added
Table 105 <code>flex.participant.deletions.enumerate</code> returned data, page 86	<code>sipReasonHeader</code>	Added
<code>flex.participant.sendUserMessage</code> , page 102	<code>position</code>	Modified default from 5 to 2
Table 126 videoRx stream struct members, page 96	<code>streamType</code>	Added
Table 126 videoRx stream struct members, page 96	<code>height</code> , <code>width</code> , <code>jitter</code> , <code>framesReceived</code> , <code>frameErrors</code> , <code>frameRate</code>	Modified
Table 127 videoTx stream struct members, page 97	<code>streamType</code>	Added
Table 127 videoTx stream struct members, page 97	<code>height</code> , <code>width</code> , <code>configuredBitRate</code> , <code>frameRate</code>	Modified
Table 128 contentVideoRx stream struct members, page 97	<code>streamType</code> , and marked as always Transcoded	Added
Table 129 contentVideoTx stream struct members, page 98	<code>streamType</code> , and marked as always Transcoded	Added
<code>system.info</code> , page 114	<code>depHash</code>	Added

**Table 178 API version 4.0(2.8) change summary**

XML-RPC Request / Topic	Parameter	Change
<code>callHome.configure</code> , page 43	<code>mode</code> , <code>automatic</code>	New command
<code>callHome.query</code> , page 44	<code>mode</code> , <code>automatic</code>	New command
<code>system.info</code> [p.1]	<code>cpuModel</code> <code>cpuCount</code> <code>cpuAvx</code>	Added

**Table 179 API version 4.0 change summary**

XML-RPC Request / Topic	Parameter	Change
<a href="#">device.feature.add, page 46</a>	<code>key</code>	New command
<a href="#">device.feature.remove, page 46</a>	<code>key</code>	New command
<a href="#">device.query, page 50</a>	<code>mediaResourceRestarts, key, expiry</code>	Added
<a href="#">device.query, page 50</a>	<code>currentTime, restartTime, uptime</code>	Documentation corrected
<a href="#">Fault Codes, page 122</a>	61, 204	New fault codes
<a href="#">system.info, page 114</a>	<code>softwareVersion</code>	Added
<a href="#">Enumerated Types, page 26</a>	<ul style="list-style-type: none"> <li>■ <a href="#">Table 18 Cascade roles enumerated type, page 29</a></li> <li>■ <a href="#">Table 19 Optimization profiles enumerated type, page 29</a></li> <li>■ <a href="#">Enumerated Types, page 26</a></li> <li>■ <a href="#">Table 22 Participant connection state enumerated type, page 30</a></li> <li>■ <a href="#">Table 23 Encryption status enumerated type, page 30</a></li> <li>■ <a href="#">Enumerated Types, page 26</a></li> </ul>	Added
<a href="#">Call Attributes Struct, page 34</a>	<code>recordingDeviceIndicateOnly, displayLayoutSwitchingMode, indicateMuting, allowStarSixMuting</code>	Added
<a href="#">Call Attributes Struct, page 34</a>	<code>videoRxFlowControlOnViewedSize</code>	When <code>true</code> , behavior modified. Flow control now only requested when received stream not viewed by other participants.
<a href="#">flex.conference.create, page 57</a> , <a href="#">flex.conference.query, page 72</a> , <a href="#">flex.conference.modify, page 67</a>	<code>optimizationProfile, useCustomMutedCanUnmuteMessage, customMutedCanUnmuteMessage, useCustomMutedCannotUnmuteMessage, customMutedCannotUnmuteMessage</code>	Added
<a href="#">flex.conference.create, page 57</a> , <a href="#">flex.conference.query, page 72</a> , <a href="#">flex.conference.modify, page 67</a>	<code>voiceSwitchingSensitivity</code>	Default modified from 50 to 60.
<a href="#">flex.conference.enumerate, page 64</a>	<code>conferenceName, presenterID, importantID</code>	Added
<a href="#">flex.conference.status, page 78</a>	<code>presenterID</code>	Added
<a href="#">flex.participant.enumerate, page 87</a>	<code>displayName, connectionState</code>	Added
<a href="#">flex.participant.advanced.enumerate, page 80</a>	Added <code>encryptionStatus, layout, mediaStatus, rxBandwidth, txBandwidth, and protocol</code>	New command. Alternative to <code>flex.participant.enumerate</code>

**Table 179 API version 4.0 change summary (continued)**

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.participant.deletions.enumerate</a> , page 86	<code>conferenceID</code>	Added
<a href="#">flex.participant.create</a> , page 84, <a href="#">flex.participant.query</a> , page 91	<code>cascadeRole</code>	Added
<a href="#">flex.participant.status</a> , page 103	<code>layout</code>	Added
<a href="#">Feedback Events</a> , page 24	<code>flexParticipantAdvancedEnum</code>	New feedback event
<a href="#">system.xml on Virtual Machine</a> , page 121		New reference topic

**Table 180 API version 3.1 change summary**

XML-RPC Request / Topic	Parameter	Change
<a href="#">system.info</a> , page 114	<code>clusterType</code>	Added
<a href="#">device.query</a> , page 50	<code>activatedLicenses</code>	Added
<a href="#">Enumerated Types</a> , page 26	Audio gain modes ( <code>gainModeDisabled</code> , <code>gainModeAutomatic</code> , <code>gainModeFixed</code> )  Control levels ( <code>controlNone</code> , <code>controlLocal</code> , <code>controlConference</code> )	Added
<a href="#">Call Attributes Struct</a> , page 34	<code>audioReceiveGainMode</code> , <code>deferConnect</code> , <code>alwaysReconnect</code> , <code>displayForceDefaultLayout</code> , <code>iXEnabled</code>	Added
<a href="#">Call Attributes Struct</a> , page 34	<code>audioReceiveGain</code>	Modified. Previously, <code>audioReceiveGain</code> was always applied. In 3.1, <code>audioReceiveGain</code> is ignored unless the <code>audioReceiveGainMode</code> is <code>gainModeFixed</code> .
<a href="#">Call Attributes Struct</a> , page 34	<code>maxTransmitPacketSize</code>	Description modified.
<a href="#">Participant Call Definition Struct</a> , page 40	Default values	Documented
<a href="#">Fault Codes</a> , page 122 reference topic		Documented
<a href="#">flex.conference.create</a> , page 57, <a href="#">flex.conference.modify</a> , page 67, <a href="#">flex.conference.query</a> , page 72	<code>conferenceDescription</code> , <code>chairControlLevel</code> , <code>guestControlLevel</code>	Added
<a href="#">flex.conference.create</a> , page 57, <a href="#">flex.conference.modify</a> , page 67, <a href="#">flex.conference.query</a> , page 72	<code>welcomeMessageScreen</code>	Modified
<a href="#">flex.participant.create</a> , page 84, <a href="#">flex.participant.query</a> , page 91, <a href="#">flex.participant.sendDTMF</a> , page 101	<code>dtmf</code>	Modified



**Table 180 API version 3.1 change summary (continued)**

XML-RPC Request / Topic	Parameter	Change
<a href="#">flex.participant.deletions.enumerate</a> , <a href="#">page 86</a>	<code>extended</code> , <code>IDs</code> , <code>participantID</code> , <code>conferenceID</code>	Added
<a href="#">flex.participant.media.enumerate</a> , <a href="#">page 89</a>	<code>conferenceID</code>	Added
<a href="#">flex.participant.requestDiagnostics</a> , <a href="#">page 93</a>	<code>clearPathOverhead</code> , <code>clearPathRecovered</code> , <code>packetsLost</code> , <code>clearPathLTRF</code> , <code>clearPathLTRFRepaired</code>	Added



## Cisco Legal Information

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies are considered un-Controlled copies and the original on-line version should be referred to for latest version.

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

© 2016 Cisco Systems, Inc. All rights reserved.

## Cisco Trademark

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

