Cisco TelePresence Server API

Product Programming Reference Guide

First Published: August 2016

4.4

Contents

Contents 3
Overview 7
Flexible Operation Mode 8
Introduction 8
Remotely Managed Mode API Change Summary 8
Design Considerations 9
XML-RPC Implementation 9
Transport 10
Encoding 10
Message Flow 10
Data Types and Sizes 11
HTTP keep-alives 12
API Overview 12
Authentication 12
Identifiers and Client References 13
Conference URI Identifiers 13
Participants 15
" Unlimited" Integers 16
Media Credits 17
Media Reservation 18
Enumeration 18
DTMF 20
Feedback Receivers 20
Data structures and enumerated types 22
Enumerated Types 22
Structs 29
Flexible API Command Reference 39
callHome.configure 40
callHome.query 41
cdrlog.enumerate 41
cdrlog.query 43
device.feature.add 43
device.feature.remove 43
device.health.query 43
device.network.modify 44
device.network.query 45
device.qos.modify 47
device.qos.query 48
device.query 48
device.restart 49

device.restartlog.query 49
device.time.modify 50
device.time.query 50
feedbackReceiver.configure 51
feedbackReceiver.query 51
feedbackReceiver.reconfigure 52
feedbackReceiver.remove 52
feedbackReceiver.status 53
flex.call.status 53
flex.conference.create 54
flex.conference.deletions.enumerate 61
flex.conference.destroy 62
flex.conference.enumerate 62
flex.conference.getMetadata 64
flex.conference.modify 65
flex.conference.query 69
flex.conference.sendUserMessage 75
flex.conference.sendWarning 76
flex.conference.status 76
flex.licenseMode.modify 77
flex.licenseMode.verify 78
flex.participant.advanced.enumerate 78
flex.participant.call.disconnect 81
flex.participant.clearImportant 81
flex.participant.create 81
flex.participant.deletions.enumerate 83
flex.participant.destroy 84
flex.participant.enumerate 85
flex.participant.media.enumerate 87
flex.participant.modify 89
flex.participant.query 89
flex.participant.requestDiagnostics 90
flex.participant.requestPreview 97
flex.participant.sendDTMF 99
flex.participant.sendUserMessage 100
flex.participant.setImportant 100
flex.participant.setMute 100
flex.participant.status 101
flex.reservation.create (experimental only—do not use) 104
flex.reservation.query (experimental only—do not use) 104
flex.reservation.modify (experimental only—do not use) 105
flex.reservation.status (experimental only—do not use) 105
flex.reservation.enumerate (experimental only—do not use) 105

flex.reservation.deletions.enumerate (experimental only—do not use) 106
flex.resource.query 106
flex.resource.status 108
logs.protocols.modify 109
logs.protocols.query 109
logs.syslog.modify 110
logs.syslog.query 110
services.modify 110
services.query 111
sip.modify 112
sip.query 113
system.info 113
user.create 115
user.destroy 115
user.modify 116
user.enumerate 116
Related Information 117
system.xml on 8710 and 7010 117
system.xml on Media 310/320 119
system.xml on Virtual Machine 120
Fault Codes 121
Example XML-RPC Response to flex.conference.create 124
Remotely Managed API Change History 125
Cisco Legal Information 137
Cisco Trademark 137

Overview

This guide describes the API available in version 4.4 of Cisco TelePresence Server. Version 4.4 or later only supports flexible operation mode (remotely managed).

Flexible Operation Mode, page 8 describes the API available in 4.4. 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 release 4.3 onwards 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 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 4.4 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
- Cisco Meeting Server 1000

Remotely Managed Mode API Change Summary

The latest Cisco TelePresence Server API is version 4.4. 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 125.

Table 1 API version 4.4 change summary

XML-RPC Request / Topic	Parameter	Change
flex.conference.create, page 54	`audioJoinNotification` `audioLeaveNotification`	New parameters
flex.conference.modify, page 65	`audioJoinNotification` `audioLeaveNotification`	New parameters
flex.conference.query, page 69	`audioJoinNotification` `audioLeaveNotification`	New parameters
flex.conference.create, page 54	`displayAudioAvatarMode`	New parameters
flex.conference.query, page 69	`displayAudioAvatarMode`	New parameters
flex.conference.sendUserMessage, page 75	`message`	Increased to 500 text characters maximum

Table 1 API version 4.4 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
flex.participant.sendUserMessage, page 100	`message`	Increased to 500 text characters maximum
Table 30 Audio Avatar mode enumerated type, page 29	`all` `preferVideo`	New enumerated type
Participant Call Definition Struct, page 37	`remoteAddress`	String increased from 80 to 1024 characters for Cisco TelePresence Server on Virtual Machine.
Table 10 Single screen layout enumerated type, page 24	`layoutOnePlusN`	New OnePlusN layout setting for single screen participants.
Table 33 callAttributes struct members, page 32	`multistreamMode`	Default changed to `multistreamOff`.

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 39.
  - 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 121.

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>
            <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
`authenticationUser`	string	**Required**. User name.
`authenticationPassword`	string	**Required**. 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 81 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 13.

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 81 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 >= 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	
	name	type	value
0 to 2147483647	`duration`	integer	0 to 2147483647
infinity / unlimited	`durationUnlimited`	boolean	`true`

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 106.

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
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 87.

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 61 with no parameters other than the Authentication, page 12 parameters, and store the `cookie` string parameter returned as the *deletions cookie*.
2. Invoke flex.conference.enumerate, page 62 with no parameters other than authentication parameters.
3. Go to step 5.
4. Invoke flex.conference.enumerate, page 62 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 61 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 61 is invoked before flex.conference.enumerate, page 62. This ensures that the deletion of any conference returned by flex.conference.enumerate, page 62 will be returned by flex.conference.deletions.enumerate, page 61 when it occurs.

It is also possible that flex.conference.deletions.enumerate, page 61 will return the IDs of conferences which have not been returned by flex.conference.enumerate, page 62. This can happen when a conference is created and destroyed before flex.conference.enumerate, page 62 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 85 and/or flex.participant.media.enumerate, page 87
- flex.participant.deletions.enumerate, page 83

Feedback Events and Enumeration

Feedback events are generated to aid incremental enumeration of conferences and participants. See Feedback Events, page 21 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 62 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 62.
- When a conference is destroyed, the `flexConferenceDeletionsEnum` event is generated. Invoke flex.conference.deletions.enumerate, page 61 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 85 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 85 method.
- When a participant is created or its media resource state changes, the `flexParticipantMediaEnum` event is generated. Invoke flex.participant.media.enumerate, page 87 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 87 method.
- When a participant is destroyed, the `flexParticipantDeletionsEnum` event is generated. Invoke flex.participant.deletions.enumerate, page 83 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 51 to register a receiver to listen for one or more Feedback Events, page 21.
- Use feedbackReceiver.query, page 51 to return a list of receivers that are configured on the device.
- Use feedbackReceiver.reconfigure, page 52 to change the configuration of an existing feedback receiver.
- Use feedbackReceiver.remove, page 52 to remove an existing feedback receiver.
- Use feedbackReceiver.status, page 53 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 **device.query** 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 When to consider a TelePresence Server to be no longer alive.
flexConferenceDeletionsEnum	One or more conferences have been destroyed. Invoke flex.conference.deletions.enumerate, page 61 to get the identifiers of conferences destroyed.
flexConferenceEnum	The state of one or more conferences has changed. Invoke flex.conference.enumerate, page 62 to get the changes.

Table 6 Feedback events (continued)

Event	Description
flexResourceConfiguration	The resource configuration has changed. Media blades have been added or removed. Invoke flex.resource.query, page 106 to retrieve the new configuration.
flexParticipantAdvancedEnum	Participants have been created or their state has changed. Invoke flex.participant.advanced.enumerate, page 78 to get the changes.
flexParticipantDeletionsEnum	Participants have been destroyed. Invoke flex.participant.deletions.enumerate, page 83 to get the identifiers of participants destroyed.
flexParticipantEnum	Participants have been created or their state has changed. Invoke flex.participant.enumerate, page 85 to get the changes.
flexParticipantMediaEnum	Participants have been created or their media resources state has changed. Invoke flex.participant.media.enumerate, page 87 to get the changes.
flexResourceStatus	Resource usage has changed: calls, participants, conferences, media tokens, and media credits. Invoke flex.resource.status, page 108 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 108; 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 23
- Table 8 Motion vs Sharpness enumerated type, page 23
- Table 9 Aspect ratio enumerated type, page 24
- Table 10 Single screen layout enumerated type, page 24
- Table 11 Multi-screen layout enumerated type, page 24
- Table 12 Call protocol enumerated type, page 24
- Table 13 Video format enumerated type, page 24
- Table 14 Participant encryption enumerated type, page 24
- Table 15 Video transmit size enumerated type, page 25
- Table 16 Call conference state enumerated type, page 25
- Table 17 Call state enumerated type, page 25
- Table 18 Cascade roles enumerated type, page 25
- Table 19 Optimization profiles enumerated type, page 26
- Table 20 Switching mode enumerated type, page 26
- Table 21 Multistream mode enumerated type, page 26
- Table 22 Participant connection state enumerated type, page 27
- Table 23 Encryption status enumerated type, page 27
- Table 24 Media Status enumerated type, page 27
- Table 25 Full screen mode enumerated type, page 28
- Table 26 Audio gain mode enumerated type, page 28
- Table 27 Control level enumerated type, page 28
- Table 28 Participant role enumerated type (experimental only—do not use), page 28
- Table 29 Resource optimization mode enumerated type, page 29
- Table 30 Audio Avatar mode enumerated type, page 29

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.
`layoutOnePlusN`	The OnePlusN layout family. One larger pane nestled in up to 12 smaller panes.

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
`NTSC`	National Television System Committee (NTSC) video format, fractions of 30 fps.
`PAL`	Phase Alternating Line (PAL) video format, fractions of 25 fps.

Table 14 Participant encryption enumerated type

Name	Description
`forbidden`	Encryption is denied.
`required`	Encryption is mandatory.
`optional`	Encryption is supported but not mandatory.

Table 15 Video transmit size enumerated type

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

Table 16 Call conference state enumerated type

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

Table 17 Call state enumerated type

Name	Description
`callStateIdle`	Call is inactive.
`callStateAlerting`	Endpoint is ringing.
`callStateAnswering`	Call is in the process of being connected.
`callStateConnected`	Endpoint is connected.
`callStateRetrying`	Call is being retried.
`callStateFailed`	Call has failed all retries **Note**: 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

`cascadeRole` value	Description
`cascadeNone`	The participant is not a cascade link.

Table 18 Cascade roles enumerated type (continued)

cascadeRole value	Description
cascadeMaster	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.
cascadeSlave	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

optimizationProfile value	Description
maximizeEfficiency	Screen licenses are conserved aggressively. This value gives the most calls for the available resources.
favorEfficiency	This is a balance of efficiency and experience that favors conserving screen licenses over attempting to grant the requested resolution.
favorExperience	Default. This is a balance of efficiency and experience that favors granting the requested resolution over conserving screen licenses.
maximizeExperience	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 optimizationProfile to capabilitySetOnly), calls will be capable of higher resolutions at lower bandwidths but the inefficiency in allocation could well outweigh the benefit.
capabilitySetOnly	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

displayLayoutSwitchingMode value	Description
switchingRoomSwitched	Room switching is used. When a participant from a room speaks, all segments (panels) from the room are shown on three-screen endpoints.
switchingSegmentSwitched	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

Value	Description
multistreamOff	Multistream not supported.
multistreamOn	Multistream supported.

Table 22 Participant connection state enumerated type

`connectionState` value	Description
`disconnected`	All calls for this participant are disconnected
`connecting`	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.
`connected`	All the participant's calls are fully connected and active in the conference.
`onHold`	The participant's calls are all connected, but the participant is on hold (not sending or receiving media)
`disconnecting`	The participant is currently being shown the exit lobby before it is disconnected
`neverConnected`	Participant has at least one configured URI and has never received any calls
`deferred`	Participant has deferConnect enabled and is waiting for other calls
`partiallyFailed`	One or more outgoing calls have failed all retry attempts
`audioReceiverFailed`	The outgoing audio receiver call has failed on a participant that only has outgoing calls **Note**: 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

`encryptionStatus` value	Description
`unknown`	The encryption status is not yet known because it is still being negotiated
`encrypted`	All of the associated channels are encrypted
`mixed`	Some of the associated channels are encrypted, others are unencrypted
`unencrypted`	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 mutli-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 `allowed`. 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
`any`	Any participant.
`accessLevelChair`	A participant with an `accessLevel` of `chair`.
`accessLevelGuest`	A participant with an `accessLevel` of `guest`.
`cascade`	A participant with a `cascadeRole` of `cascadeMaster` or `cascadeSlave`.

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

Table 29 Resource optimization mode enumerated type

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

Table 30 Audio Avatar mode enumerated type

Name	Description
`all`	All audio participants are replaced by a generic avatar icon which is presented in views as if it were input from their camera. Many avatars may be visible simultaneously. Avatars are full screen when the participant is the current loudest speaker.
`preferVideo`	Audio participants are replaced by a generic avatar icon when presented in views as above. No more than one such avatar is visible in any view. Where possible the avatar is presented in the film strip rather than full-screen, this increases the screen area used for participant video.

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 39.

- Media Tokens Struct, page 29
- Participant Media Resources Struct, page 30
- Call Attributes Struct, page 31
- Conference URI Details Struct, page 36
- Participant Call Definition Struct, page 37
- Enumerated Reservation Information Struct (experimental only—do not use), page 38

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 31 Media tokens struct members

Parameter name	Type	Description
`total`	integer (>= 0)	Required. Maximum total media token usage permitted across all channels.
`maxPerChannel`	integer (>= 0, <= `total`)	Maximum media resource usage permitted for a channel. Default: see `maxPerChannelUnlimited`. Note: `maxPerChannel` must be less than `total`.
`maxPerChannelUnlimited`	boolean	Whether an unlimited number of resources can be used for a channel. See "Unlimited" Integers, page 16. Default: `true`.

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 106 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 32 Participant media resources struct members

Parameter name	Type	Description
`mediaTokensMainVideo`	struct	Media token values representing the maximum resources that can be assigned to main video within a participant. See Media Tokens Struct, page 29.
`mediaTokensExtendedVideo`	struct	Media token values representing the maximum resources that can be assigned to extended video within a participant. See Media Tokens Struct, page 29.
`mediaTokensAudio`	struct	Media token values representing the maximum resources that can be assigned to audio within a participant. See Media Tokens Struct, page 29.
`numMediaCredits`	integer (>= 0)	Number of credits configured for the participant. See Media Credits, page 17.

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 17.

`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 33 callAttributes struct members, page 32:

- Participant specification (see flex.participant.create, page 81 command).
- Participant modification (see flex.participant.modify, page 89 command).
- Conference URI specification (seeConference URI Details Struct, page 36).
- Conference creation (see flex.conference.create, page 54 command).
- Conference modification (see flex.conference.modify, page 65 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 33 callAttributes struct members, page 32) 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 33 `callAttributes` struct members

Parameter name	Type	Description
`accessLevel`	string	Access level associated with this participant, when connected. See Table 7 Access level enumerated type, page 23. Default: `chair`. **Note:** Incoming calls are reported as `unknown` until they have been authorized (reach the lobby screen).
`encryption`	string	Encryption setting. See Table 14 Participant encryption enumerated type, page 24. Default: `optional`.
`autoDisconnect`	boolean	Whether this call automatically disconnects if the only calls connected to a conference have `autoDisconnect` set. Default: `false`.
`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. Note: the old range is still accepted for backwards compatibility only.

Table 33 callAttributes struct members (continued)

Parameter name	Type	Description
`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%).
`videoRxFlowControlOnErrors`	boolean	Flow control on video errors. Default: `true`.
`videoRxFlowControlOnViewedSize`	boolean	Flow control based on viewed size. Default: `true`. This parameter is modified in version 4.0 as follows: if `true`, 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.
`videoTxSizeOptimization`	string	Video transmit size setting. See Table 15 Video transmit size enumerated type, page 25. Default: `dynamicCodecAndResolution`.
`presentationContributionAllowed`	boolean	Whether the endpoint can contribute presentations to a conference. Default: `true`.
`presentationTakeoverAllowed`	boolean	Whether the endpoint can start contributing presentations even if the conference already has an active presentation. Default: `true`.
`videoTxPresentationAllowed`	boolean	Whether the endpoint can receive presentations in its extended video channel (if available). Default: `true`.
`videoTxPresentationMainVideoAllowed`	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 `false`. Setting it to `true` will result in it implicitly being set to `false`. Any implicit changes will be reflected in the return values of flex.participant.query, page 89. Default: `true`.
`audioStereoEnabled`	boolean	Whether support for stereo is enabled. Default: `true`.
`audioDirectionalEnabled`	boolean	Whether directional audio is enabled. Default: `true`.
`indicateUnencryptedParticipants`	boolean	Whether the unencrypted icon is displayed. Default: `true`.
`indicateAudioOnlyParticipants`	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: `true`.
`mainVideoTxPictureAspectRatio`	string	Permissible aspect ratios for the main video channel. See Table 9 Aspect ratio enumerated type, page 24. Default: `onlySixteenToNine`.

Table 33 callAttributes struct members (continued)

Parameter name	Type	Description
`extendedVideoTxPictureAspectRatio`	string	Permissible aspect ratios for the extended video channel. See Table 9 Aspect ratio enumerated type, page 24. Default: `allowAllResolutions`.
`videoTxFormat`	string	Video format. See Table 13 Video format enumerated type, page 24. Default: `NTSC`.
`videoTxMotionSharpness`	string	Motion sharpness setting. See Table 8 Motion vs Sharpness enumerated type, page 23. 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 Table 25 Full screen mode enumerated type, page 28. 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 Table 10 Single screen layout enumerated type, page 24. Default: `layoutActivePresence`. **Note**: This has no effect on multistream calls.
`displayDefaultLayoutMultiScreen`	string	Layout scheme used for multiscreen endpoints. See Table 11 Multi-screen layout enumerated type, page 24. Default: `layoutActivePresence`. **Note**: 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`. **Note**: This has no effect on multistream calls.
`displayShowEndpointNames`	boolean	Whether endpoint names are shown as panel labels. Default: `true`.
`audioReceiveGainMode`	string	See Table 26 Audio gain mode enumerated type, page 28. Either `gainModeDisabled`, `gainModeAutomatic`, or `gainModeFixed`. Default: `gainModeAutomatic`.

Table 33 callAttributes struct members (continued)

Parameter name	Type	Description
`audioReceiveGain`	integer (-12000 to +12000)	Gain for received audio when `audioReceiveGainMode` is `gainModeFixed`. The unit is millidecibels and the default value is `0`. This call attribute is not required for other values of `audioReceiveGainMode` and will be ignored if supplied in these cases.
`audioTransmitGain`	integer (-12000 to +12000)	Gain for transmitted audio, measured in millidecibels. Default: 0.
`forceTIP`	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: `false`. 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.
`audioRxStartMuted`	boolean	Whether audio from the endpoint is muted at the start of a call. Default: `false`.
`videoRxStartMuted`	boolean	Whether video from the endpoint is muted at the start of a call. Default: `false`.
`audioTxStartMuted`	boolean	Whether audio to the endpoint is muted at the start of a call. Default: `false`.
`videoTxStartMuted`	boolean	Whether video to the endpoint is muted at the start of a call. Default: `false`.
`autoReconnect`	boolean	Whether outgoing calls dropped for abnormal reasons are reconnected. Not effective for incoming calls. This setting will be ignored if `alwaysReconnect` is `true`. Default: `false`.
`recordingDevice`	boolean	Whether this call is treated as a recording device by muting received video and displaying a recording icon on other endpoints. Default: `false`.
`recordingDeviceIndicateOnly`	boolean	Whether this call is indicated as a recording device by displaying a recording icon on other endpoints. Received video is not muted. Default: `false`.
`deferConnect`	boolean	Applies only to calls dialed out from the TelePresence Server. If `true`, the TelePresence Server defers connecting this call until at least one other call is connected to the conference. If `false`, the TelePresence Server connects the call as soon as the conference starts. Default: `false`.

Table 33 callAttributes struct members (continued)

Parameter name	Type	Description
`alwaysReconnect`	boolean	If `true`, the TelePresence Server will reconnect this call in all disconnection circumstances. Applies only to calls dialed out from the TelePresence Server. Default: `false`. **Caution:** 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: `true`.
`displayLayoutSwitchingMode`	string	Defines how the display of this multicamera system is switched into view on three-screen endpoints. Either `switchingRoomSwitched` or `switchingSegmentSwitched`. Defaults to `switchingSegmentSwitched`. See Enumerated Types, page 22. **Note**: 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. `true` shows the icon when the participant has been muted. Default: `true`.
`allowStarSixMuting`	boolean	Defines whether participants can mute or unmute their audio by pressing *6. `true` allows the participant to use the *6 combination to mute/unmute. Default: `true`.
`multistreamMode`	string	Select if multistream is used for the call. see Table 21 Multistream mode enumerated type, page 26. Default is `multistreamOff`.
`displayOnScreenMessages`	boolean	Display on screen messages. Default: `true`.

Conference URI Details Struct

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

Table 34 Conference Connection Definitions

Parameter name	Type	Description
`URI`	string (80)	**Required**. String used by endpoints to connect to this conference. See Conference URI Identifiers, page 13.

Table 34 Conference Connection Definitions (continued)

Parameter name	Type	Description
`callBandwidth`	integer (>= `minCallBandwidth` and <= `maxCallBandwidth`)	**Required**. Connection bandwidth measured in bits per second. See flex.resource.query, page 106.
`PIN`	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.
`callAttributes`	struct	This Call Attributes Struct, page 31 contains attributes applied to calls connecting to a conference using the aforementioned URI. See Call Attributes Struct, page 31 and How Call Attributes are Derived, page 31. Default: inherits the conference default call attributes, see flex.conference.create, page 54 and flex.conference.modify, page 65.
`participantMediaResources`	struct	The Participant Media Resources Struct, page 30 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 35 Incoming participant call definition struct members

Parameter name	Type	Description
`URI`	string (80)	**Required**. String used to connect to this conference. See Participant Conference URIs, page 15.
`callBandwidth`	integer (>= `minCallBandwidth` and <= `maxCallBandwidth`)	**Required**. Connection bandwidth measured in bits per second. See flex.resource.query, page 106.
`disconnectOnIncoming`	boolean	Whether the existing call (if one exists) is disconnected when an incoming call comes through the `URI`. Default: `false`.

Table 36 Outgoing participant call definition struct members

Parameter name	Type	Description
`remoteAddress`	string	**Required**. Address of the endpoint expressed in the form of an endpoint address or E164 number. Up to 1024 characters for Cisco TelePresence Server on Virtual Machine, up to 80 characters for all other TelePresence Server platforms.
`protocol`	string	**Required**. Call control protocol for outgoing call only. See Enumerated Types, page 22.
`callBandwidth`	integer (>= `minCallBandwidth` and <= `maxCallBandwidth`)	**Required**. Connection bandwidth in bits per second. See flex.resource.query, page 106.
`toOverride`	string (80)	**Optional**. 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 37 Incoming participant PIN definition

Parameter name	type	Description
`PIN`	String	**Required**. PIN for this participant URI, up to 40 characters long.
`callAttributes`	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 Call Attributes Struct, page 31.

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

Table 38 Enumerated reservation information struct members

Parameter name	Type	Description
`reservationID`	string (50)	Reservation identifier, assigned by the DSC.
`reservationReference`	string (50)	Client reference string.
`conferenceID`	string (50)	Identifier of the conference the reservation is associated with.
`participantRole`	string	What role the reservation is for.
`numParticipants`	integer (>=1)	Number of participants the reservation is for.
`maxQuality`	integer (>=0)	Maximum quality (in total credits) of each participant.
`totalReservedCredits`	integer (>=0)	Total amount of credits reserved by the reservation.
`numParticipantsRemaining`	integer (>=0)	Number of participant still capable of using the reservation.
`numCreditsRemaining`	integer (>=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 40
- callHome.query, page 41
- cdrlog.enumerate, page 41
- cdrlog.query, page 43
- device.feature.add, page 43
- device.feature.remove, page 43
- device.health.query, page 43
- device.network.modify, page 44
- device.network.query, page 45
- device.query, page 48
- device.restart, page 49
- device.restartlog.query, page 49
- device.time.modify, page 50
- device.time.query, page 50
- feedbackReceiver.configure, page 51
- feedbackReceiver.query, page 51
- feedbackReceiver.reconfigure, page 52
- feedbackReceiver.remove, page 52
- feedbackReceiver.status, page 53
- flex.call.status, page 53
- flex.conference.create, page 54
- flex.conference.deletions.enumerate, page 61
- flex.conference.destroy, page 62
- flex.conference.enumerate, page 62
- flex.conference.getMetadata, page 64
- flex.conference.modify, page 65
- flex.conference.query, page 69
- flex.conference.sendUserMessage, page 75
- flex.conference.sendWarning, page 76
- flex.conference.status, page 76
- flex.licenseMode.modify, page 77
- flex.licenseMode.verify, page 78
- flex.participant.advanced.enumerate, page 78

- flex.participant.call.disconnect, page 81
- flex.participant.clearImportant, page 81
- flex.participant.create, page 81
- flex.participant.deletions.enumerate, page 83
- flex.participant.destroy, page 84
- flex.participant.enumerate, page 85
- flex.participant.media.enumerate, page 87
- flex.participant.modify, page 89
- flex.participant.query, page 89
- flex.participant.requestDiagnostics, page 90
- flex.participant.requestPreview, page 97
- flex.participant.sendDTMF, page 99
- flex.participant.sendUserMessage, page 100
- flex.participant.setImportant, page 100
- flex.participant.setMute, page 100
- flex.participant.status, page 101
- flex.resource.query, page 106
- flex.resource.status, page 108
- logs.syslog.modify, page 110
- logs.syslog.query, page 110
- services.modify, page 110
- services.query, page 111
- sip.modify, page 112
- sip.query, page 113
- system.info, page 113
- user.create, page 115
- user.destroy, page 115
- user.enumerate, page 116
- user.modify, page 116

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 39 `callHome.configure` **inputs**

Parameter name	Type	Description
`mode`	string	Set the Call Home mode. One of `disabled` or `anonymous`. Can only be set to `anonymous` if the encryption feature key is present. Defaults to `disabled` if it has never been configured. Omit the parameter to leave the current setting unchanged.
`automatic`	boolean	Controls automatic Call Home. `true` enables automatic Call Home. `false` disables automatic Call Home. Only has effect when mode is `anonymous`. 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 40 `callHome.query` **returned data**

Parameter name	Type	Description
`mode`	string	Call Home mode. One of `disabled` or `anonymous`. Defaults to `disabled` if it has never been configured.
`automatic`	boolean	`true` if automatic Call Home is enabled. `false` if automatic Call Home is disabled. Only has effect if mode is `anonymous`. Defaults to `false` 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 41 `cdrlog.enumerate` **optional or conditional inputs**

Parameter name	Type	Description
`index`	integer	Index from which to get events. The device returns the `nextIndex` so the application can use it to retrieve the next enumeration of CDR data. If `index` 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.

Table 41 cdrlog.enumerate optional or conditional inputs (continued)

Parameter name	Type	Description
`numEvents`	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 `eventsRemaining` parameter in the response and re-enumerate, starting from `nextIndex`, if necessary.
`filter`	array	An array of strings, which contain the names of event types by which to filter the response. Omit `filter` to return all event types or include a subset of the following: `conferenceStarted`, `conferenceFinished`, `conferenceActive`, `conferenceInactive`, `participantConnected`, `participantJoined`, `participantMediaSummary`, `participantLeft`, `participantDisconnected`.

Table 42 `cdrlog.enumerate` returned data

Parameter name	Type	Description
`startIndex`	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 `startIndex` with the index provided gives the number of dropped records.
`nextIndex`	integer	Revision number of the data being provided, reusable in a subsequent call to the API.
`eventsRemaining`	boolean	If `true`, there is more data in the requested enumeration than has been returned in this response.
`currentTime`	dateTime.iso8601	The system's current time (UTC).
`events`	array of structs	Each member of the array is a struct that represents a recorded event. The structures all have some common fields (`time`, `type`, `index`) 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 43 Common CDR log event parameters

Parameter name	Type	Description
`time`	dateTime.iso8601	Date and time when the event was logged; for example, `20110119T13:52:42`.
`type`	string	Name of the event type.
`index`	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 44 `cdrlog.query` **returned data**

Parameter name	Type	Description
`firstIndex`	integer	Index of the oldest stored event.
`numEvents`	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 45 `device.feature.add` **inputs**

Parameter name	Type	Description
`key`	string	**Required**. 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 46 `device.feature.remove` **inputs**

Parameter name	Type	Description
`key`	string	**Required**. 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 47 `device.health.query` **returned data**

Parameter name	Type	Description
`cpuLoad`	integer	CPU load expressed as a percentage of the maximum.
`fanStatus`	string	`ok` or `outOfSpec`. 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 47 device.health.query returned data (continued)

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

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 48 `device.network.modify` inputs

Parameter name	Type	Description
`portA`	struct	**Optional**. The struct contains configuration and status information for Ethernet port A on the device. See Table 49 Port struct members., page 45.
`dns`	array of structs	**Optional**. Each member of the array is a struct representing a set of DNS parameters for the device. See Table 50 DNS struct members, page 45.

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 49 Port struct members.

Parameter name	Type	Description
`ipv4Enabled`	boolean	True if IPv4 interface is enabled.
`dhcpv4`	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.
`ipv4Address`	string	IPv4 address in the dotted quad format.
`ipv4SubnetMask`	string	IPv4 subnet mask in the dotted quad format.
`defaultIpv4Gateway`	string	IPv4 address in the dotted quad format.
`ipv6Enabled`	boolean	Whether IPv6 interface is enabled. Always returned unless there are no IP interfaces enabled on the port (neither IPv4 nor IPv6 is enabled).
`ipv6Address`	string	IPv6 address in CIDRformat. Not returned if not configured.
`ipv6Conf`	string	Indicates how the IPv6 address is assigned. One of `automatic` (IPv6 address is configured by SLAAC/DHCPv6) or `manual` (IPv6 address is configured manually). Not returned if not configured.
`ipv6PrefixLength`	integer	Length of the IPv6 address prefix.
`defaultIpv6Gateway`	string	Address of the IPv6 default gateway in CIDR format.

Table 50 DNS struct members

Parameter name	Type	Description
`dnsConfiguration`	string	How the TelePresence Server is acquires its name server address. One of `portAIPv4`, `portAIPv6`, or `manual`. If `manual`, then at least one name server must be provided.
`hostName`	string	Host name of the device.
`nameServer`	string	IP address of the name server, in dotted quad format (IPv4) or CIDR format (IPv6).
`nameServerSecondary`	string	IP address of the secondary name server, in dotted quad format (IPv4) or CIDR format (IPv6).
`domainName`	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 51 `device.network.query` returned data

Parameter name	Type	Description
`portA`	struct	The struct contains configuration and status information for Ethernet port A on the device. See Table 52 Port struct members, page 46.
`dns`	array of structs	Each member of the array is a struct representing a set of DNS parameters for the queried device. See Table 53 DNS struct members, page 47.

Table 52 Port struct members

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

Table 52 Port struct members (continued)

Parameter name	Type	Description
`dhcpv4`	boolean	Whether the ipv4 address is allocated by DHCP. Not returned if not configured.
`ipv4Address`	string	IPv4 address in the dotted quad format. Not returned if not configured.
`ipv4SubnetMask`	string	IPv4 subnet mask in the dotted quad format. Not returned if not configured.
`defaultipv4Gateway`	string	IPv4 address in the dotted quad format. Not returned if not configured.
`ipv6Address`	string	IPv6 address in CIDRformat. Not returned if not configured.
`ipv6Conf`	string	Indicates how the IPv6 address is assigned. One of `automatic` (IPv6 address is configured by SLAAC/DHCPv6) or `manual` (IPv6 address is configured manually). Not returned if not configured.
`ipv6PrefixLength`	integer	Length of the IPv6 address prefix. Not returned if not configured.
`defaultIpv6Gateway`	string	Address of the IPv6 default gateway in CIDR format. Not returned if not configured.
`linkLocalIpv6Address`	string	Link local IPv6 address in CIDR format. Not returned if not configured.
`linkLocalIpv6PrefixLength`	integer	Length of the link local IPv6 address prefix. Not returned if not configured.

Table 53 DNS struct members

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

device.qos.modify

Modifies the Quality of Service (DSCP precedence) values used for all media.

Table 54 `device.qos.modify` inputs

Parameter name	Type	Description
`ipv4Audio`	Integer	From 0 to 63, used as DSCP precedence value.
`ipv4Video`	Integer	From 0 to 63, used as DSCP precedence value.
`ipv6Audio`	Integer	From 0 to 63, used as DSCP precedence value.
`ipv6Video`	Integer	From 0 to 63, used as DSCP precedence value.

device.qos.query

Returns the Quality of Service (DSCP precedence) values used for all media.

Table 55 `device.qos.query` returned data

Parameter name	Type	Description
`ipv4Audio`	Integer	From 0 to 63, used as DSCP precedence value.
`ipv4Video`	Integer	From 0 to 63, used as DSCP precedence value.
`ipv6Audio`	Integer	From 0 to 63, used as DSCP precedence value.
`ipv6Video`	Integer	From 0 to 63, used as DSCP precedence value.

device.query

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

Table 56 `device.query` returned data

Parameter name	Type	Description
`currentTime`	dateTime.iso8601	The system's current date and time.
`restartTime`	dateTime.iso8601	The system's date and time when it started.
`uptime`	integer	The difference, in seconds, between the system's current time and the system's restart time.
`serial`	string	Serial number of this device.
`apiVersion`	string	Version number of the API implemented by this TelePresence Server.
`activatedFeatures`	array of structs	Each member of the array is a struct, representing an active feature. See Table 57 Active feature struct members, page 48.
`activatedLicenses`	array of structs	Each member of the array is a struct, representing an active license. See Table 58 Active license struct members, page 49.
`shutdownStatus`	string	Displays one of the following: **`notShutdown`**, **`shutdownInProgress`**, **`shutdown`**, or **`error`**.
`mediaResourceRestarts`	integer	The count of unexpected restarts that have occurred on the device's media resources (signal processor chips).

Table 57 Active feature struct members

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

Table 58 Active license struct members

Parameter name	Type	Description
`license`	string	The name of the license.
`ports`	integer	The number of screen licenses provided by this license.
`key`	string	The unique code associated with the license.
`expiry`	dateTime.iso8601	The time at which this temporary key will expire. `expiry` 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 59 `device.restart` input parameters

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

device.restartlog.query

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

Table 60 `device.restartlog.query` returned data

Parameter name	Type	Description
`log`	array of structs	Each member of the array is a struct containing a restart `reason`. See Table 61 Log struct members, page 49. This information source is called "system log" in the web interface.

Table 61 Log struct members

Parameter name	Type	Description
`time`	dateTime.iso8601	Date and time of the restart.
`reason`	string	Reason for the device restart. See Table 62 Restart reason enumerated type, page 49.

Table 62 Restart `reason` enumerated type

`reason` 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.

Table 62 Restart reason enumerated type (continued)

`reason` value	Description
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 63 `device.time.modify` inputs

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

device.time.query

Returns the current time settings of the device.

Table 64 `device.time.query` returned data

Parameter name	Type	Description
`currentTime`	date Time ISO8601	The current time (UTC).
`ntpEnabled`	boolean	True if NTP is enabled.
`utcOffsetHours`	Integer	Range -12 to +14 inclusive. Together with `utcOffsetMinutes` specifies offset to UTC.

Table 64 device.time.query returned data (continued)

Parameter name	Type	Description
`utcOffsetMinutes`	Integer	Range 0 to 59 inclusive. Together with `utcOffsetHours` specifies offset to UTC.
`ntpHost`	string	DNS or IP address of an NTP server.
`ntpStatus`	string	The NTP client's current status; one of `disabled`, `synchronizing`, `synchronized` or `error`.

feedbackReceiver.configure

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

Table 65 `feedbackReceiver.configure` inputs

Parameter name	Type	Description
`receiverURI`	string (255)	**Required**. Fully-qualified `http` or `https` URI (for example, `http://tms1:8080/RPC2`) to which feedback events are sent. If no port number is specified, the device uses the protocol defaults (80 and 443 respectively).
`receiverIndex`	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. **Note:** The default `receiverIndex` 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.
`sourceIdentifier`	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.
`subscribedEvents`	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 Feedback Events, page 21. If this array is absent, the receiver subscribes to all notifications by default. Default: all events.

Table 66 `feedbackReceiver.configure` returned data

Parameter name	Type	Description
`receiverIndex`	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 67 `feedbackReceiver.query` **returned data**

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

Table 68 Feedback receiver struct members

Parameter name	Type	Description
index	integer (1–20)	Position of this feedback receiver in the table of feedback receivers. The index number is also the feedback receiver ID.
sourceIdentifier	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.
receiverURI	string (255)	Fully-qualified **http** or **https** URI (for example, **http://tms1:8080/RPC2**) 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 69 `feedbackReceiver.reconfigure` **inputs**

Parameter name	Type	Description
receiverIndex	integer (1–20)	**Required**. Index of the feedback receiver to be reconfigured. The call returns a fault if there is no feedback receiver at the specified **receiverIndex**.
receiverURI	string (255)	Fully-qualified **http** or **https** URI (for example, **http://tms1:8080/RPC2**) to which feedback events are sent. If omitted, the device uses the originally configured **receiverURI**.
sourceIdentifier	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 **sourceIdentifier**.
subscribedEvents	array	Array of strings identifying the events to which the receiver subscribes. See Feedback Events, page 21. 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 70 `feedbackReceiver.remove` **inputs**

Parameter name	Type	Description
receiverIndex	integer (1–20)	**Required**. 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 71 `feedbackReceiver.status` **inputs**

Parameter name	Type	Description
`receiverIndex`	integer (1–20)	**Required**. Index of the feedback receiver.

Table 72 `feedbackReceiver.status` **returned data**

Parameter name	Type	Description
`receiverIndex`	integer (1–20)	Index of the feedback receiver entry, which also serves as the feedback receiver ID.
`sourceIdentifier`	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.
`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 Feedback Events, page 21.

flex.call.status

Returns the status of the specified call.

Table 73 `flex.call.status` **inputs**

Parameter name	Type	Description
`callID`	string (50)	**Required**. Call identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

Table 74 `flex.call.status` **returned data**

Parameter name	Type	Description
`callID`	string (50)	Call identifier. See Identifiers and Client References, page 13.
`conferenceID`	string (50)	Identifier of the conference to which the call is connected or is in the process of connecting. See Identifiers and Client References, page 13.
`conferenceState`	string	State of call connection to the conference. See Table 16 Call conference state enumerated type, page 25.
`callState`	string	State of the call. See Table 17 Call state enumerated type, page 25.
`incoming`	boolean	Direction of the call: `true` indicates incoming; `false` indicates outgoing.

Table 74 flex.call.status returned data (continued)

Parameter name	Type	Description
`protocol`	string	Call control protocol. See Table 12 Call protocol enumerated type, page 24.
`address`	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.
`participantID`	string (50)	Participant identifier. See Identifiers and Client References, page 13. 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 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 75 `flex.conference.create` inputs

Parameter name	Type	Description
`participantMediaResources`	struct	Required. This Participant Media Resources Struct, page 30 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.

Table 75 flex.conference.create inputs (continued)

Parameter name	Type	Description
`conferenceReference`	string (50)	Conference client reference string. See Identifiers and Client References, page 13. Default: empty.
`conferenceName`	string (80)	Human readable label for the conference. Default: empty string.
`conferenceDescription`	string (500)	Human readable description of the conference. If set, the TelePresence Server uses this value for ActiveControl's conference description. Default: empty string.
`URIs`	array of structs	Each member of the array is aTable 34 Conference Connection Definitions, page 36 defining a unique conference URI and its associated access levels and media tokens. 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. Default: empty array. **Note**: The URI does not need to be unique, if certain conditions are met. See URIs Do Not Need to be Unique, page 15
`conferenceMediaTokens`	integer (>= 0)	Maximum number of media tokens that can be used for the conference. Do not supply `conferenceMediaTokens` if you supply `conferenceMediaTokensUnlimited`. This will result in a fault. Default: Absent. See `conferenceMediaTokensUnlimited`.
`conferenceMediaTokensUnlimited`	boolean	Whether no limit is defined for the number of media tokens that can be used for the conference. Do not supply `conferenceMediaTokensUnlimited` if you supply `conferenceMediaTokens`. This will result in a fault. See "Unlimited" Integers, page 16. Default: `true`.
`conferenceMediaCredits`	integer (>= 0)	Maximum number of media credits that can be used for the conference. Do not supply `conferenceMediaCredits` if you supply `conferenceMediaCreditsUnlimited`. This will result in a fault. Default: Absent. See `conferenceMediaCreditsUnlimited`.

Table 75 flex.conference.create inputs (continued)

Parameter name	Type	Description
`conferenceMediaCreditsUnlimited`	boolean	Whether no limit is defined for the number of media credits that can be used for the conference. Do not supply `conferenceMediaCreditsUnlimited` if you supply `conferenceMediaCredits`. This will result in a fault. See "Unlimited" Integers, page 16. Default: `true`.
`waitForChair`	boolean	Whether callers must wait for a chair to join the conference. Default: `true`.
`disconnectOnChairExit`	boolean	Whether callers are disconnected when the last chair leaves the conference. Default: `false`.
`terminateWithLastCall`	boolean	Whether the conference is destroyed with the last call. Default: `false`.
`locked`	boolean	Whether the conference is locked, causing incoming calls to be rejected. Default: `false`.
`startTime`	integer (>= 0)	Number of seconds to wait before starting the conference. Default: 0.
`duration`	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 Conference send warning. Do not supply `duration` if you supply `durationUnlimited`. This will result in a fault. Default: Absent. See `durationUnlimited`.
`durationUnlimited`	boolean	Whether an unlimited duration is assigned for the conference. If this field is present and the value is `true`, the value for the conference duration is ignored. Do not supply `durationUnlimited` if you supply `duration`. This will result in a fault. See "Unlimited" Integers, page 16. Default: `true`.
`billingCode`	string (80)	Billing code string. Default: empty.
`callAttributes`	struct	Conference default call attributes. See How Call Attributes are Derived, page 31. Default: see Table 33 callAttributes struct members, page 32 for defaults of struct members.

Table 75 flex.conference.create inputs (continued)

Parameter name	Type	Description
`maxParticipants`	integer (>= 0)	Maximum number of participants that can connect to this conference. Do not supply `maxParticipants` if you supply `maxParticipantsUnlimited`. This will result in a fault. Default: Absent. See `maxParticipantsUnlimited`.
`maxParticipantsUnlimited`	boolean	Whether an unlimited number of participants are allowed to connect to this conference. Do not supply `maxParticipantsUnlimited` if you supply `maxParticipants`. This will result in a fault.See "Unlimited" Integers, page 16. Default: `true`.
`voiceSwitchingSensitivity`	integer (0–100)	Voice switching sensitivity. Default: 60.
`welcomeScreen`	boolean	Whether a welcome screen message is displayed for 5 seconds when a call joins the conference. See `welcomeScreenMessage` for contents of the message. Default: `true`.
`welcomeScreenMessage`	string (500)	Welcome screen message for this conference. If this string is empty, the conference name is displayed as the welcome screen message. If `conferenceDescription` is not set, the TelePresence Server uses this value for ActiveControl's conference description. Default: empty string.
`useCustomPINEntryMessage`	boolean	Whether a custom message is displayed in the PIN entry screen. Default: `false`.
`customPINEntryMessage`	string (200)	Custom message for PIN entry. Only used if `useCustomPINEntryMessage` is `true`. Default: empty.
`useCustomOptionalPINEntryMessage`	boolean	Display custom message at shared PIN entry screen when conference can be entered without a PIN. Default: `false`.
`customOptionalPINEntryMessage`	String (200)	Custom message for shared PIN entry screen when conference can be entered without a PIN, if `useCustomOptionalPINEntryMessage` is `true`. Default: empty.
`useCustomPINIncorrectMessage`	boolean	Whether a custom warning message is displayed in the PIN entry screen after an incorrect PIN has been entered. Default: `false`.
`customPINIncorrectMessage`	string (100)	Custom warning message for incorrect PIN entry. Only used if `useCustomPINIncorrectMessage` is `true`. Default: empty.

Table 75 flex.conference.create inputs (continued)

Parameter name	Type	Description
`useCustomWaitingForChairMessage`	boolean	Whether a custom message is displayed when waiting for a chair to join the conference. Default: `false`.
`customPINEntryFailedMessage`	string (200)	Optional. Custom message displayed on the exit lobby if a participant is disconnected for failing to enter the PIN three times.
`useCustomPINEntryFailedMessage`	boolean	Optional. Whether the custom message is displayed for the PIN failure exit lobby.
`customDisconnectPlatitudeMessage`	string (200)	Optional. Custom "thank you" message displayed on all exit lobbies.
`useCustomDisconnectPlatitudeMessage`	boolean	Optional. Whether the custom "thank you" message is displayed on all exit lobbies.
`customWaitingForChairMessage`	string (500)	Custom message displayed when waiting for a chair to join the conference. Only used if `useCustomWaitingForChairMessage` is `true`. Default: empty.
`use CustomOnlyVideoParticipantMessage`	boolean	Whether a custom message is displayed when a participant is the only (active) video participant. Default: `false`.
`customOnlyVideoParticipantMessage`	string (500)	Custom message displayed when a participant is the only (active) video participant. Only used if `useCustomOnlyVideoParticipantMessage` is `true`. Default: empty.
`useCustomConferenceEndingMessage`	boolean	Whether a custom message is displayed when the conference is about to end. Default: `false`.
`customConferenceEndingMessage`	string (100)	Custom message displayed when the conference is about to end. Only used if `useCustomConferenceEndingMessage` is `true`. Default: empty.
`metadata`	base64 (<= 512 bytes)	Client meta data. Default: zero length.
`unlockWithLastCall`	boolean	Whether the conference is unlocked when the participant leaves the conference. Default: `true`.
`guestControlLevel`	string	See Table 27 Control level enumerated type, page 28. Either `controlNone`, `controlLocal`, or `controlConference`. Defines the level of control to which the guests in this conference are entitled. Default: `controlLocal`.

Table 75 flex.conference.create inputs (continued)

Parameter name	Type	Description
`chairControlLevel`	string	See Table 27 Control level enumerated type, page 28. Either `controlNone`, `controlLocal`, or `controlConference`. Defines the level of control to which the chairpersons in this conference are entitled. Default: `controlConference`.
`optimizationProfile`	string	Sets the optimization profile for the conference, which defines how the conference reports far-end token values for endpoints.See Table 19 Optimization profiles enumerated type, page 26. Default: `favorExperience`.
`useCustomMutedCanUnmuteMessage`	boolean	`true` 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: `true`.
`customMutedCanUnmuteMessage`	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 `useCustomMutedCanUnmuteMessage` is `false`. Default: Empty.
`useCustomMutedCannotUnmuteMessage`	boolean	`true` 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: `true`.
`customMutedCannotUnmuteMessage`	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 `useCustomMutedCannotUnmuteMessage` is `false`. Default: Empty.
`exitScreen`	boolean	Display a user friendly message when an endpoint is disconnected from a conference in an orderly manner. Default: `true`.
`useCustomConferenceEndedExitMessage`	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 `exitScreen` is `true`. Default: `false`.
`customConferenceEndedExitMessage`	string (200)	Custom message displayed when a participant has been disconnected, if `useCustomConferenceEndedExitMessage` is `true`. Default: empty.

Table 75 flex.conference.create 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.
`customWelcomeScreenAudio`	string (80)	Optional.The url of a file to use as the voice prompt on the conference welcome screen.
`useCustomWelcomeScreenAudio`	boolean	Optional. Use the custom welcome screen voice prompt.
`customPINEntryAudio`	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.
`useCustomPINEntryAudio`	boolean	Optional. Use the custom PIN entry voice prompt.
`customOptionalPINEntryAudio`	string (80)	Optional. The url of a file to use as the voice prompt on the PIN entry screen when the PIN is optional
`useCustomOptionalPINEntryAudio`	boolean	Optional. Use the custom optional PIN entry voice prompt.
`customPINIncorrectAudio`	string (80)	Optional. The url of a file to use as a custom voice prompt for incorrect PIN entry.
`useCustomPINIncorrectAudio`	boolean	Optional. Use the custom voice prompt for incorrect PIN
`customConferenceEndedExitAudio`	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.
`useCustomConferenceEndedExitAudio`	boolean	Optional. Use the custom voice prompt for conference ending.
`customParticipantDisconnectedExitAudio`	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.
`useCustomParticipantDisconnectedExitAudio`	boolean	Optional. Use the custom voice prompt for participant disconnection.
`customPINFailedExitAudio`	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.
`useCustomPINFailedExitAudio`	boolean	Optional. Use the custom voice prompt for participant disconnection on PIN failure.
`customWaitingForChairAudio`	string (url)	Optional. The url of a file to use as a custom voice prompt when waiting for the conference host.

Table 75 flex.conference.create inputs (continued)

Parameter name	Type	Description
`useCustomWaitingForChairAudio`	boolean	Optional. Use the custom voice prompt for waiting for the conference host.
`customOnlyParticipantAudio`	string (80)	Optional. The url of a file to use as a custom voice prompt when there is only one participant in the conference.
`useCustomOnlyParticipantAudio`	boolean	Optional. Use the custom voice prompt for only participant.
`resourceOptimizationMode`	string (30)	Optional. Controls how resources in the conference are optimized. Default: `byParticipant`.
`customBackgroundImageURL`	string (80)	Optional. The url of a file to use as a custom background on lobby screens. Default: empty.
`audioJoinNotification`	string	Optional. Controls the notification received when participants join a conference. One of `none` or `all`. Default: `none`.
`audioLeaveNotification`	string	Optional. Controls the notification received when participants leave a conference. One of `none` or `all`. Default: `none`.
`displayAudioAvatarMode`	string	Optional. Controls the presentation of audio avatars to video participants in the conference . One of `all` or `preferVideo`. Default: `preferVideo`.

Table 76 `flex.conference.create` returned data

Parameter name	Type	Description
`conferenceID`	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 Identifiers and Client References, page 13.
`conferenceReference`	string (50)	Conference client reference string. Returned if string length > 0. See Identifiers and Client References, page 13.

flex.conference.deletions.enumerate

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

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

Parameter name	Type	Description
`cookie`	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.
`max`	integer (> 0)	Maximum number of conference deletion records to return in response. If `max` is not specified, as many records are returned as is possible.

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

Parameter name	Type	Description
`moreAvailable`	boolean	Whether there are more conference deletions to be enumerated.
`cookie`	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
`conferenceIDs`	array of identifiers	The identifiers of conferences that have been deleted. See Identifiers and Client References, page 13.

flex.conference.destroy

Destroys the specified conference. No parameters are returned.

Table 79 `flex.conference.destroy` **inputs**

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`allowExitScreen`	boolean	**Optional**. If `allowExitScreen` is `false` then, the exitScreen setting for the conference is overridden and set `false`. The exit lobby is not displayed and participant is disconnected immediately. Default: `False`

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 18.

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 80 `flex.conference.enumerate` **inputs**

Parameter name	Type	Description
`cookie`	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.
`max`	integer (> 0)	Maximum number of conference details to return in response. If `max` is not specified, as many records will be returned as is possible.

Table 81 `flex.conference.enumerate` **returned data**

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

Table 81 flex.conference.enumerate returned data (continued)

Parameter name	Type	Description
`cookie`	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
`conferences`	array of structs	Each member of the array is a struct representing a single conference. See Table 82 Conference information struct, page 63. The array is returned empty if there are no conferences to enumerate.
`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).
`conferenceCreditsRequired` (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 `maxMediaCredits`.
`conferenceCreditsUsed` (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 82 Conference information struct

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`conferenceReference`	string (50)	Conference client reference string. Returned if string length > 0. See Identifiers and Client References, page 13.
`conferenceName`	string (80)	Human readable label for the conference. Returned if string length > 0.
`locked`	boolean	Whether the conference is locked.
`active`	boolean	Whether the conference has started.
`terminating`	boolean	Whether the conference has been extended for the duration of the exit lobby.
`numParticipants`	integer (>= 0)	Number of participants connected to the conference, including participants with incoming calls that have not yet been established.
`callTokensConfiguredMainVideo`	integer (>= 0)	Number of tokens configured for main video summed over all participants connected to the conference.
`callTokensAllocatedMainVideo`	integer (>= 0)	Number of tokens allocated for main video summed over all participants connected to the conference.

Table 82 Conference information struct (continued)

Parameter name	Type	Description
`callTokensConfiguredExtendedVideo`	integer (>= 0)	Number of tokens configured for extended video summed over all participants connected to the conference.
`callTokensAllocatedExtendedVideo`	integer (>= 0)	Number of tokens allocated for extended video summed over all participants connected to the conference.
`callTokensConfiguredAudio`	integer (>= 0)	Number of tokens configured for audio summed over all participants connected to the conference.
`callTokensAllocatedAudio`	integer (>= 0)	Number of tokens allocated for audio summed over all participants connected to the conference.
`callQualityCanImproveMainVideo`	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.
`callQualityCanImproveExtendedVideo`	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.
`callQualityCanImproveAudio`	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.
`creditsConfigured`	integer (>= 0)	Number of configured credits summed over all participants connected to the conference.
`creditsAllocated`	integer (>= 0)	Number of allocated credits summed over all participants connected to the conference.
`presenterID`	string (50)	The identifier of the participant who is currently presenting to the conference. Not returned if no participant is presenting.
`importantID`	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 flex.participant.setImportant, page 100 and flex.participant.clearImportant, page 81.

flex.conference.getMetadata

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

Table 83 `flex.conference.getMetadata` inputs

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

Table 84 `flex.conference.getMetadata` **returned data**

Parameter name	Type	Description
`metadata`	base64 (<= 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 16:

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

Table 85 `flex.conference.modify` **inputs**

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`conferenceReference`	string (50)	Conference client reference string. See Identifiers and Client References, page 13.
`conferenceName`	string (80)	Human readable label for the conference.
`conferenceDescription`	string (500)	Human readable description of the conference. If set, the TelePresence Server uses this value for ActiveControl's conference description.
`URIs`	array of structs	Each member of the array is a Table 34 Conference Connection Definitions, page 36 that defines conference URIs, associated access levels, and media resources. 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. 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. **Note**: The URI does not need to be unique, if certain conditions are met. See URIs Do Not Need to be Unique, page 15

Table 85 flex.conference.modify inputs (continued)

Parameter name	Type	Description
`conferenceMediaTokens`	integer (>= 0)	Maximum number of media tokens that can be used for the conference. See `conferenceMediaTokensUnlimited`.
`conferenceMediaTokensUnlimited`	boolean	Whether no limit is defined for the number of media tokens that can be used for the conference.
`conferenceMediaCredits`	integer (>= 0)	Maximum number of media credits that can be used for the conference. See `conferenceMediaCreditsUnlimited`.
`conferenceMediaCreditsUnlimited`	boolean	Whether no limit is defined for the number of media credits that can be used for the conference.
`waitForChair`	boolean	Whether callers must wait for a chair to join the conference.
`disconnectOnChairExit`	boolean	Whether callers are disconnected when the last chair leaves the conference.
`terminateWithLastCall`	boolean	Whether the conference is destroyed with the last call.
`locked`	boolean	Whether the conference is locked causing incoming calls to be rejected.
`duration`	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 Conference send warning. It is an error to set the duration to a value that requires it to have ended in the past. See `durationUnlimited`.
`durationUnlimited`	boolean	Whether an unlimited duration is assigned for the conference.
`billingCode`	string (80)	Billing code string.
`callAttributes`	struct	This Call Attributes Struct, page 31 defines the conference default call attributes. See How Call Attributes are Derived, page 31.
`participantMediaResources`	struct	This Participant Media Resources Struct, page 30 struct defines the conference default participant media resource configuration. These settings can be overridden by those defined at the `Conference URI` or participant level. All members of the struct are changed; unspecified optional fields are set to their default values.
`maxParticipants`	integer (>= 0)	Maximum number of participants that can connect to this conference. See `maxParticipantsUnlimited`.
`maxParticipantsUnlimited`	boolean	Whether an unlimited number of participants are allowed to connect to this conference. See "Unlimited" Integers, page 16.

Table 85 flex.conference.modify inputs (continued)

Parameter name	Type	Description
`voiceSwitchingSensitivity`	integer (0–100)	Voice switching sensitivity. Default: 60.
`welcomeScreen`	boolean	Whether a welcome screen message is displayed for 5 seconds when a call joins the conference. See `welcomeScreenMessage` for the contents of the message.
`welcomeScreenMessage`	string (500)	Welcome screen message for this conference. If this message is empty, the conference name is displayed as the welcome screen message. If `conferenceDescription` is not set, the TelePresence Server uses this value for ActiveControl's conference description.
`useCustomPINEntryMessage`	boolean	Whether a custom message is displayed in the PIN entry screen.
`customPINEntryMessage`	string (200)	Custom message for PIN entry. Only used if `useCustomPINEntryMessage` is `true`.
`useCustomOptionalPINEntryMessage`	boolean	Display custom message at shared PIN entry screen when conference can be entered without a PIN
`customOptionalPINEntryMessage`	string (200)	Custom message for shared PIN entry screen when conference can be entered without a PIN, if `useCustomOptionalPINEntryMessage` is `true`.
`useCustomPINIncorrectMessage`	boolean	Whether a custom warning message is displayed in the PIN entry screen after an incorrect PIN has been entered.
`customPINIncorrectMessage`	string (100)	Custom warning message for incorrect PIN entry. Only used if `useCustomPINIncorrectMessage` is `true`.
`useCustomWaitingForChairMessage`	boolean	Whether a custom message is displayed when waiting for a chair to join the conference.
`customWaitingForChairMessage`	string (500)	Custom message displayed when waiting for a chair to join the conference. Only used if `useCustomWaitingForChairMessage` is `true`.
`customPINEntryFailedMessage`	string (200)	Optional. Custom message displayed on the exit lobby if a participant is disconnected for failing to enter the PIN 3 times.
`useCustomPINEntryFailedMessage`	boolean	Optional. Whether the custom message is displayed for the PIN failure exit lobby.
`customDisconnectPlatitudeMessage`	string (200)	Optional. Custom "thank you" message displayed on all exit lobbies.
`useCustomDisconnectPlatitudeMessage`	boolean	Optional. Whether the custom "thank you" message is displayed on all exit lobbies.
`use CustomOnlyVideoParticipantMessage`	boolean	Whether a custom message is displayed when a participant is the only (active) video participant.

Table 85 flex.conference.modify inputs (continued)

Parameter name	Type	Description
`customOnlyVideoParticipantMessage`	string (500)	Custom message displayed when a participant is the only (active) video participant. Only used if `useCustomOnlyVideoParticipantMessage` is `true`.
`useCustomConferenceEndingMessage`	boolean	Whether a custom message is displayed when the conference is about to end.
`customConferenceEndingMessage`	string (100)	Custom message displayed when the conference is about to end. Only used if `useCustomConferenceEndingMessage` is `true`.
`metadata`	base64 (<= 512 bytes)	Client metadata.
`unlockWithLastCall`	boolean	Whether the conference is unlocked when the participant leaves the conference.
`guestControlLevel`	string	See Table 27 Control level enumerated type, page 28. Either `controlNone`, `controlLocal`, or `controlConference`. Defines the level of control to which the guests in this conference are entitled.
`chairControlLevel`	string	See Table 27 Control level enumerated type, page 28. Either `controlNone`, `controlLocal`, or `controlConference`. Defines the level of control to which the chairpersons in this conference are entitled.
`optimizationProfile`	string	Sets the optimization profile for the conference, which defines how the conference reports far-end token values for endpoints.See Table 19 Optimization profiles enumerated type, page 26. Default: `favorExperience`.
`useCustomMutedCanUnmuteMessage`	boolean	`true` 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: `true`.
`customMutedCanUnmuteMessage`	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 `useCustomMutedCanUnmuteMessage` is `false`. Default: Empty.
`useCustomMutedCannotUnmuteMessage`	boolean	`true` 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: `true`.

Table 85 flex.conference.modify inputs (continued)

Parameter name	Type	Description
`customMutedCannotUnmuteMessage`	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 `useCustomMutedCannotUnmuteMessage` is `false`. Default: Empty.
`exitScreen`	boolean	Display a user friendly message when an endpoint is disconnected from a conference in an orderly manner. Default: `true`.
`useCustomConferenceEndedExitMessage`	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 `exitScreen` is `true`. Default: `false`
`customConferenceEndedExitMessage`	string (200)	Custom message displayed when a participant has been disconnected, if `useCustomConferenceEndedExitMessage` is `true`. Default: empty
`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
`audioJoinNotification`	string	Optional. Controls the notification received when participants join a conference. One of `none` or `all`. Default: `none`.
`audioLeaveNotification`	string	Optional. Controls the notification received when participants leave a conference. One of `none` or `all`. Default: `none`.

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 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 86 `flex.conference.query` **inputs**

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

Table 87 `flex.conference.query` **returned data**

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`URIs`	array of structs	Each member of the array is a Table 34 Conference Connection Definitions, page 36 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 Identifiers and Client References, page 13.
`conferenceName`	string (80)	Human readable label for the conference. Returned if string length > 0.
`conferenceDescription`	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.
`conferenceMediaTokens`	integer (>= 0)	Maximum number of media tokens that can be used for the conference. See `conferenceMediaTokensUnlimited`. Not returned if it has not been set.
`conferenceMediaTokensUnlimited`	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 "Unlimited" Integers, page 16.
`conferenceMediaCredits`	integer (>= 0)	Maximum number of media credits that can be used for the conference. See `conferenceMediaCreditsUnlimited`. Not returned if it has not been set.
`conferenceMediaCreditsUnlimited`	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 "Unlimited" Integers, page 16.

Table 87 flex.conference.query returned data (continued)

Parameter name	Type	Description
`duration`	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 Conference send warning. Not returned if it has not been set. See `durationUnlimited`.
`durationUnlimited`	boolean	Whether an unlimited duration is assigned for the conference. Not returned if it has not been set. See "Unlimited" Integers, page 16.
`billingCode`	string (80)	Billing code string. Not returned if it has not been set.
`maxParticipants`	integer (>= 0)	Maximum number of participants that can connect to this conference. Not returned if it has not been set. See `maxParticipantsUnlimited`.
`maxParticipantsUnlimited`	boolean	Whether an unlimited number of participants are allowed to connect to this conference. Not returned if it has not been set. See "Unlimited" Integers, page 16.
`waitForChair`	boolean	Whether callers must wait for a chair to join the conference.
`disconnectOnChairExit`	boolean	Whether callers are disconnected when the last chair leaves the conference.
`terminateWithLastCall`	boolean	Whether the conference is destroyed with the last call.
`locked`	boolean	Whether the conference is locked so that only outgoing calls are permitted.
`startTime`	integer (>= 0)	Number of seconds after which to start the conference.
`callAttributes`	struct	This Call Attributes Struct, page 31 defines the conference default call attributes. See How Call Attributes are Derived, page 31.
`voiceSwitchingSensitivity`	integer (0–100)	Voice switching sensitivity.
`participantMediaResources`	struct	This Participant Media Resources Struct, page 30 defines the conference default participant media resource configuration. These settings can be over-ridden by those defined at the `Conference URI` or participant level.
`welcomeScreen`	boolean	Whether a welcome screen message is displayed for 5 seconds when a call joins the conference. See `welcomeScreenMessage` for contents of the message.

Table 87 flex.conference.query returned data (continued)

Parameter name	Type	Description
`welcomeScreenMessage`	string (500)	Welcome screen message for this conference. If this message is empty, the conference name is displayed as the welcome screen message. If `conferenceDescription` is not set, the TelePresence Server uses this value for ActiveControl's conference description.
`useCustomPINEntryMessage`	boolean	Whether a custom message is displayed in the PIN entry screen.
`customPINEntryMessage`	string (200)	`Custom message for PIN entry. Only used if useCustomPINEntryMessage is true.`
`useCustomOptionalPINEntryMessage`	boolean	Display custom message at shared PIN entry screen when conference can be entered without a PIN
`customOptionalPINEntryMessage`	string (200)	Custom message for shared PIN entry screen when conference can be entered without a PIN, if `useCustomOptionalPINEntryMessage` is `true`.
`useCustomPINIncorrectMessage`	boolean	Whether a custom warning message is displayed in the PIN entry screen after an incorrect PIN has been entered.
`customPINIncorrectMessage`	string (100)	Custom warning message for incorrect PIN entry, Only used if `useCustomPINIncorrectMessage` is `true`.
`useCustomWaitingForChairMessage`	boolean	Whether a custom message is displayed when waiting for a chair to join the conference.
`customWaitingForChairMessage`	string (500)	Custom message displayed when waiting for a chair to join the conference. Only used if `useCustomWaitingForChairMessage` is `true`.
`useCustomOnlyVideoParticipantMessage`	boolean	Whether a custom message is displayed when the participant is the only (active) video participant.
`customOnlyVideoParticipantMessage`	string (500)	Custom message displayed when the participant is the only (active) video participant. Only used if `useCustomOnlyVideoParticipantMessage` is `true`.
`useCustomConferenceEndingMessage`	boolean	Whether a custom message is displayed when the conference is about to end.
`customConferenceEndingMessage`	string (100)	Custom message displayed when the conference is about to end. Only used if `useCustomConferenceEndingMessage` is `true`.
`hasMetadata`	boolean	Whether the conference has non-zero-length metadata.
`unlockWithLastCall`	boolean	Whether the conference is unlocked when the last participant leaves the conference.

Table 87 flex.conference.query returned data (continued)

Parameter name	Type	Description
`guestControlLevel`	string	See Table 27 Control level enumerated type, page 28. Either `controlNone`, `controlLocal`, or `controlConference`. Defines the level of control to which the guests in this conference are entitled.
`chairControlLevel`	string	See Table 27 Control level enumerated type, page 28. Either `controlNone`, `controlLocal`, or `controlConference`. Defines the level of control to which the chairpersons in this conference are entitled.
`optimizationProfile`	string	The optimization profile for the conference, which defines how the conference reports far-end token values for endpoints.See Table 19 Optimization profiles enumerated type, page 26.
`useCustomMutedCanUnmuteMessage`	boolean	`true` 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.
`customMutedCanUnmuteMessage`	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 `useCustomMutedCanUnmuteMessage` is `false`.
`useCustomMutedCannotUnmuteMessage`	boolean	`true` 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.
`customMutedCannotUnmuteMessage`	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 `useCustomMutedCannotUnmuteMessage` is `false`.
`exitScreen`	boolean	Display a user friendly message when an endpoint is disconnected from a conference in an orderly manner. Default: `true`.
`useCustomConferenceEndedExitMessage`	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 `exitScreen` is `true`. Default: `false`
`customConferenceEndedExitMessage`	string (200)	Custom message displayed when a participant has been disconnected, if `useCustomConferenceEndedExitMessage` is `true`. Default: empty

Table 87 flex.conference.query returned data (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
`customPINEntryFailedMessage`	string (200)	Optional. Custom message displayed on the exit lobby if a participant is disconnected for failing to enter the PIN 3 times.
`useCustomPINEntryFailedMessage`	boolean	Optional. Whether the custom message is displayed for the PIN failure exit lobby.
`customDisconnectPlatitudeMessage`	string (200)	Optional. Custom "thank you" message displayed on all exit lobbies.
`useCustomDisconnectPlatitudeMessage`	boolean	Optional. Whether the custom "thank you" message is displayed on all exit lobbies.
`customWelcomeScreenAudio`	string (80)	Optional.The url of a file to use as the voice prompt on the conference welcome screen.
`useCustomWelcomeScreenAudio`	boolean	Optional. Use the custom welcome screen voice prompt.
`customPINEntryAudio`	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.
`useCustomPINEntryAudio`	boolean	Optional. Use the custom PIN entry voice prompt.
`customOptionalPINEntryAudio`	string (80)	Optional. The url of a file to use as the voice prompt on the PIN entry screen when the PIN is optional
`useCustomOptionalPINEntryAudio`	boolean	Optional. Use the custom optional PIN entry voice prompt.
`customPINIncorrectAudio`	string (80)	Optional. The url of a file to use as a custom voice prompt for incorrect PIN entry.
`useCustomPINIncorrectAudio`	boolean	Optional. Use the custom voice prompt for incorrect PIN
`customConferenceEndedExitAudio`	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.
`useCustomConferenceEndedExitAudio`	boolean	Optional. Use the custom voice prompt for conference ending.
`customParticipantDisconnectedExitAudio`	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.

Table 87 flex.conference.query returned data (continued)

Parameter name	Type	Description
`useCustomParticipantDisconnectedExitAudio`	boolean	Optional. Use the custom voice prompt for participant disconnection.
`customPINFailedExitAudio`	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.
`useCustomPINFailedExitAudio`	boolean	Optional. Use the custom voice prompt for participant disconnection on PIN failure.
`customWaitingForChairAudio`	string (80)	Optional. The url of a file to use as a custom voice prompt when waiting for the conference host.
`useCustomWaitingForChairAudio`	boolean	Optional. Use the custom voice prompt for waiting for the conference host.
`customOnlyParticipantAudio`	string (80)	Optional. The url of a file to use as a custom voice prompt when there is only one participant in the conference.
`useCustomOnlyParticipantAudio`	boolean	Optional. Use the custom voice prompt for only participant.
`customBackgroundImageURL`	string (url 80)	Optional. The url of a file to use as a custom background on lobby screens. Default: empty.
`audioJoinNotification`	string	Optional. Controls the notification received when participants join a conference. One of `none` or `all`. Default: `none`.
`audioLeaveNotification`	string	Optional. Controls the notification received when participants leave a conference. One of `none` or `all`. Default: `none`.
`displayAudioAvatarMode`	string	Optional. Controls the presentation of audio avatars to video participants in the conference . One of `all` or `preferVideo`. Default: `preferVideo`.

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 88 `flex.conference.sendUserMessage` inputs

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`message`	string (500)	**Required**. 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 89 `flex.conference.sendWarning` **inputs**

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`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 **not** 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 90 `flex.conference.status` **inputs**

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

Table 91 `flex.conference.status` **returned data**

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`locked`	boolean	Whether the conference is locked.

Table 91 flex.conference.status returned data (continued)

Parameter name	Type	Description
`numParticipants`	integer (>= 0)	Number of participants connected to the conference, including participants with incoming calls that have not yet been established.
`numScreens`	integer (>= 0)	Number of screens connected to the conference.
`configuredMediaTokens`	integer	Sum of configured media tokens for all calls connected to the conference.
`configuredMediaCredits`	integer	Sum of configured media credits for all calls connected to the conference.
`startTime`	integer	Amount of time, measured in seconds, between now and the conference start time. This value is < 0 for conferences that are currently active.
`active`	boolean	Whether the conference has started.
`presenterID`	string (50)	Identifier of the participant who is currently presenting to the conference. Not returned if no participant is presenting.
`conferenceReference`	string (50)	Conference client reference string. Returned if string length > 0. See Identifiers and Client References, page 13.
`endTime`	integer	Amount of time, measured in seconds, until the conference is due to end. Only returned if conference duration is limited.
`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).
`conferenceCreditsRequired` (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 `maxMediaCredits`.
`conferenceCreditsUsed` (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 92 `flex.licenseMode.modify` inputs

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

Table 93 `flex.licenseMode.modify` **returned data**

Parameter name	Type	Description
`trustSecret`	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 94 `flex.licenseMode.verify` **inputs**

Parameter name	Type	Description
No additional parameters required.		

Table 95 `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 18 and flex.participant.enumerate, page 85.

Table 96 `flex.participant.advanced.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 Identifiers and Client References, page 13. Default: none.

Table 97 `flex.participant.advanced.enumerate` **returned data**

Parameter name	Type	Description
`moreAvailable`	boolean	Whether there are more participants to be enumerated.
`cookie`	string (150)	Enumeration cookie that must be returned in the next invocation to continue the enumeration. See Enumeration, page 18.
`participants`	array of structs	Each member of the array is a struct defining a participant. See Table 98 Participant information struct members, page 79. This array may be empty.

Table 98 Participant information struct members

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`participantReference`	string (50)	Client reference string. See Identifiers and Client References, page 13. Absent if empty.
`displayName`	string (80)	The current display name for this participant. Absent if empty.
`conferenceID`	string (50)	Conference to which the participant is connected. See Identifiers and Client References, page 13.
`accessLevel`	string	Access level granted to the participant. See Table 7 Access level enumerated type, page 23. **Note:** Incoming calls are reported as `unknown` until they have been authorized (reach the lobby screen).
`connectionState`	string	One of `disconnected`, `connecting`, `connected`, or `onHold`. See Table 22 Participant connection state enumerated type, page 27.
`calls`	array of structs	Each member of this array is a struct that defines one call associated with the participant. See Table 99 Participant call information struct members, page 80. 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 `addresses` array.
`addresses`	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 Table 100 Participant address struct members, page 80. The position in the array matches that of the associated call in the `calls` array.
`encryptionStatus`	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 Table 101 encryptionStatus struct members, page 80 and Table 23 Encryption status enumerated type, page 27.
`layout`	string	The display layout that is currently shown on the participant's endpoint. One of `layoutSingle`, `layoutActivepresence`, `layoutProminent`, or `layoutEqual`. Not returned if there are no calls connected for this participant. See Table 10 Single screen layout enumerated type, page 24 or Table 11 Multi-screen layout enumerated type, page 24, depending on the type of endpoint.
`mediaStatus`	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 Table 102 mediaStatus struct members, page 81 and Enumerated Types, page 22.

Table 99 Participant call information struct members

Parameter name	Type	Description
`callID`	string (50)	Call identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`incoming`	boolean	Direction of the call: `true` indicates incoming; `false` indicates outgoing.
`address`	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.
`protocol`	string	The call control protocol used in this call. One of `h323` or `sip`. See Enumerated Types, page 22.
`rxBandwidth`	integer	
`txBandwidth`	integer	

Table 100 Participant address struct members

Parameter name	Type	Description
`URI`	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.
`remoteAddress`	string (80)	Remote address specified in flex.participant.create, page 81 to call out to the endpoint. Outgoing calls only, not returned for incoming calls. Only returned if it is not empty.

Table 101 `encryptionStatus` struct members

Parameter name	Type	Description
`callProtocol`	string	The encryption status of the call control protocol. One of `unknown`, `encrypted`, `mixed`, or `unencrypted`. See Table 23 Encryption status enumerated type, page 27.
`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 102 `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 Enumerated Types, page 22.
`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 103 `flex.participant.call.disconnect` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`position`	integer (>= 0)	**Required**. 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 104 `flex.participant.clearImportant` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

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 18)

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 37 are not met
- `cascadeRole` is not `cascadeNone` and more than one call is defined in the `calls` array

Table 105 `flex.participant.create` **inputs**

Parameter name	Type	Description
`conferenceID`	string (50)	**Required**. Conference identifier. See Identifiers and Client References, page 13.
`calls`	array of structs	**Required**. Each member of the array is a Participant Call Definition Struct, page 37 which specifes 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.
`participantReference`	string (50)	Client reference string. See Identifiers and Client References, page 13. Default: empty.
`PIN`	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. **Deprecated**: please use `PINs` instead
`PINs`	array of Participant PIN Definition, page 38	List of PINs for this participant, if any of its calls are incoming. Maximum of 2.
`callAttributes`	struct	See Call Attributes Struct, page 31 for details of struct members. The settings defined in this struct override the conference's default call attribute settings. See How Call Attributes are Derived, page 31. Default: inherits conference default call attributes.

Table 105 flex.participant.create inputs (continued)

Parameter name	Type	Description
`participantMediaResources`	struct	See Participant Media Resources Struct, page 30 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.
`camerasCrossed`	boolean	Whether cameras in the group of endpoints specified by `calls` are crossed. Ignored if the calls array length = 1. Default: `false`.
`audioIndex`	integer (>= 0)	Position in the `calls` array of the call that will receive audio. The position must exist in the `calls` array. First position is 0. Ignored if the `calls` array length = 1. Default: 0.
`contentIndex`	integer (>= 0)	Position in the `calls` array of the call that will receive content. The position must exist in the `calls` array. First position is 0. Ignored if the `calls` array length = 1. Default: 0.
`displayName`	string (80)	Configured display name for the endpoint. This overrides the endpoint display name setting. Default: empty.
`dtmf`	string (127)	Valid DTMF, page 20 characters in a sequence that is sent to the call nominated by the `audioIndex`. This sequence is only sent when dialing out and is not sent for incoming calls. Default: empty.
`callerName`	string (80)	Calling name seen by the endpoint. Not used for incoming calls. Default: empty.
`callerAddress`	string (80)	Calling address seen by the endpoint. Not used for incoming calls. Default: empty.
`cascadeRole`	string	One of `cascadeNone`, `cascadeMaster`, or `cascadeSlave`. See Table 18 Cascade roles enumerated type, page 25. Default: `cascadeNone`.

Table 106 `flex.participant.create` returned data

Parameter name	Type	Description
`participantID`	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 Identifiers and Client References, page 13.
`participantReference`	string (50)	Client reference string. Returned if not empty. See Identifiers and Client References, page 13.

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 107 `flex.participant.deletions.enumerate` **inputs**

Parameter name	Type	Description
`cookie`	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.
`max`	integer (> 0)	Maximum number of participant deletion records returned in response. If `max` is not specified, as many records are 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 Identifiers and Client References, page 13. Default: none.
`extended`	boolean	If `true`, the response includes the `IDs` array. If `false`, the response includes the `participantIDs` array. `extended` 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: `false`.

Table 108 `flex.participant.deletions.enumerate` **returned data**

Parameter name	Type	Description
`moreAvailable`	boolean	Whether there are more participant deletions to be enumerated.
`cookie`	string (150)	Cookie that must be supplied in the next invocation to continue the enumeration.
`participantIDs`	array of strings	Each member of the array is a string (50) that identifies a participant that has been deleted. See Identifiers and Client References, page 13. Not returned if `IDs` is returned.
`IDs`	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 Table 109 IDs array struct members, page 84. Not returned if `participantIDs` is returned.

Table 109 `IDs` **array struct members**

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`sipReasonHeader`	string	**Optional**. 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 110 `flex.participant.destroy` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`allowExitScreen`	boolean	**Optional**. 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 18 and flex.participant.advanced.enumerate, page 78

Table 111 `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 Identifiers and Client References, page 13. Default: none.

Table 112 `flex.participant.enumerate` **returned data**

Parameter name	Type	Description
`moreAvailable`	boolean	Whether there are more participants to be enumerated.
`cookie`	string (150)	Enumeration cookie that must be returned in the next invocation to continue the enumeration. See Enumeration, page 18.
`participants`	array of structs	Each member of the array is a struct defining a participant. See Table 113 Participant information struct members, page 85. This array may be empty.

Table 113 Participant information struct members

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`participantReference`	string (50)	Client reference string. See Identifiers and Client References, page 13. Absent if empty.
`displayName`	string (80)	The current display name for this participant. Absent if empty.

Table 113 Participant information struct members (continued)

Parameter name	Type	Description
`conferenceID`	string (50)	Conference to which the participant is connected. See Identifiers and Client References, page 13.
`accessLevel`	string	Access level granted to the participant. See Table 7 Access level enumerated type, page 23. **Note:** Incoming calls are reported as `unknown` until they have been authorized (reach the lobby screen).
`connectionState`	string	One of `disconnected`, `connecting`, `connected`, or `onHold`. See Table 22 Participant connection state enumerated type, page 27.
`calls`	array of structs	Each member of this array is a struct that defines one call associated with the participant. See Table 114 Participant call information struct members, page 86. 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 `addresses` array.
`addresses`	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 Table 115 Participant address struct members, page 86. The position in the array matches that of the associated call in the `calls` array.

Table 114 Participant call information struct members

Parameter name	Type	Description
`callID`	string (50)	Call identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`incoming`	boolean	Direction of the call: `true` indicates incoming; `false` indicates outgoing.
`address`	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.

Table 115 Participant address struct members

Parameter name	Type	Description
`URI`	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.
`remoteAddress`	string (80)	Remote address specified in flex.participant.create, page 81 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 16.

Table 116 `flex.participant.media.enumerate` inputs

Parameter name	Type	Description
`cookie`	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.
`max`	integer (> 0)	Maximum number of participant media 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 Identifiers and Client References, page 13. Default: none.

Table 117 `flex.participant.media.enumerate` returned data

Parameter name	Type	Description
`moreAvailable`	boolean	Whether there are more participants to be enumerated.
`cookie`	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
`participantMediaInfo`	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 Table 118 Participant media information struct members, page 87.

Table 118 Participant media information struct members

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`participantReference`	string (50)	Client reference string. See Identifiers and Client References, page 13. Absent if empty.
`mainVideoTokenInfo`	struct	A struct that defines the tokens for main video. See Table 119 Participant token information struct members, page 88.
`extendedVideoTokenInfo`	struct	A struct that defines the tokens for extended video (content). See Table 119 Participant token information struct members, page 88.

Table 118 Participant media information struct members (continued)

Parameter name	Type	Description
`audioTokenInfo`	struct	A struct that defines the tokens for audio. See Table 119 Participant token information struct members, page 88.
`creditsConfigured`	integer (>= 0)	Number of credits configured for the participant.
`creditsFarEnd`	integer (>= 0)	Number of credits required to match far end capability. Absent if the far-end capabilites are not yet known, or if some calls are not yet established.
`creditsNearEnd`	integer (>= 0)	Number of credits required to match near end capability. Absent if the far-end capabilites are not yet known, or if some calls are not yet established.

Table 119 Participant token information struct members

Parameter name	Type	Description
`maxTokensConfigured`	integer (>= 0)	Maximum number of tokens with respect to conference or participant configuration.
`maxTokensPerChannelConfigured`	integer (>= 0)	Maximum number of tokens per channel with respect to the conference or participant configuration. Not returned if `maxTokensPerChannelConfiguredUnlimited` is returned (`true`).
`maxTokensPerChannelConfiguredUnlimited`	boolean	Whether there is an unlimited maximum number of tokens per channel with respect to the conference or participant configuration. Not returned if `maxTokensPerChannelConfigured` is returned, in which case it is implicitly `false`.
`maxTokensPerChannelFarEnd`	integer (>= 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 capabilites are not yet known, or if some calls are not yet established.
`maxTokensFarEnd`	integer (>= 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 capabilites are not yet known, or if some calls are not yet established.
`maxTokensPerChannelNearEnd`	integer (>= 0)	Maximum number of tokens per channel, advertised by the TelePresence Server. Absent if the far-end capabilites are not yet known, or if some calls are not yet established.
`maxTokensNearEnd`	integer (>= 0)	Maximum number of tokens advertised by the TelePresence Server. Absent if the far-end capabilites 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 16.

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 120 `flex.participant.modify` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`displayName`	string (80)	Configured display name for the endpoint. This overrides the endpoint display name setting.
`callAttributes`	struct	This Call Attributes Struct, page 31 modifies the participant's call attributes. These settings override the conference default call attribute settings. See How Call Attributes are Derived, page 31.
`participantMediaResources`	struct	This Participant Media Resources Struct, page 30 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 121 `flex.participant.query` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

Table 122 `flex.participant.query` **returned data**

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`conferenceID`	string (50)	Conference identifier. See Identifiers and Client References, page 13.
`PIN`	string (40)	Numeric PIN. **Deprecated**: please use `PINs` instead

Table 122 flex.participant.query returned data (continued)

Parameter name	Type	Description
`PINs`	array of Participant PIN Definition, page 38	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 Call Attributes Struct, page 31 of the queried participant. See How Call Attributes are Derived, page 31. **Note:** The `accessLevel` for incoming calls is reported as `unknown` until they have been authorized (reached the lobby screen).
`participantMediaResources`	struct	This is the previously specified or inherited Participant Media Resources Struct, page 30 of the queried participant.
`calls`	array of structs	Each member of the array is a Participant Call Definition Struct, page 37 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 `calls` are crossed.
`audioIndex`	integer (>= 0)	Position in the `calls` array of the call that will receive audio.
`contentIndex`	integer (>= 0)	Position in the `calls` array of the call that will receive content.
`cascadeRole`	string	One of `cascadeNone`, `cascadeMaster`, or `cascadeSlave`. See Table 18 Cascade roles enumerated type, page 25.
`participantReference`	string (50)	Client reference string. See Identifiers and Client References, page 13. 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 DTMF, page 20 characters in a sequence that is sent to the call nominated by the `audioIndex`. This sequence is only sent when dialing out and is not sent for incoming calls.
`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 121.

Table 123 flex.participant.requestDiagnostics inputs

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
`receiverURI`	string (255)	**Required**. Fully-qualified `http` or `https` URI (for example, http://example.com:5050/RPC2 or https://example.com:5050/RPC2) 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 124 `flex.participant.requestDiagnostics` asynchronously returned data

Parameter name	Type	Description
`participantID`	string (50)	Identifier of the participant to which these diagnostics relate. See Identifiers and client references.
`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 Table 125 audioRx stream struct members, page 92.
`audioTx`	array	Array of `audioTx` stream structs. See Table 126 audioTx stream struct members, page 92.
`auxiliaryAudioRx`	array	Array of `auxiliaryAudioRx` stream structs. See Table 127 auxiliaryAudioRx stream struct members, page 93.
`auxiliaryAudioTx`	array	Array of `auxiliaryAudioTx` stream structs. See Table 128 auxiliaryAudioTx stream struct members, page 93.
`videoRx`	array	Array of `videoRx` stream structs. See Table 129 videoRx stream struct members, page 93.
`videoTx`	array	Array of `videoTx` stream structs. See Table 130 videoTx stream struct members, page 94.
`contentVideoRx`	array	Array of `contentRx` stream structs. See Table 131 contentVideoRx stream struct members, page 95.

Table 124 flex.participant.requestDiagnostics asynchronously returned data (continued)

Parameter name	Type	Description
`contentVideoTx`	array	Array of `contentTx` stream structs. See Table 132 contentVideoTx stream struct members, page 96.

Table 125 `audioRx` stream struct members

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

Table 126 `audioTx` stream struct members

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

Table 127 `auxiliaryAudioRx` stream struct members

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

Table 128 `auxiliaryAudioTx` stream struct members

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

Table 129 `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).

Table 129 videoRx stream struct members (continued)

Parameter name	Type	Description
`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 TelePresence Server 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 130 `videoTx` 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 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).

Table 130 videoTx stream struct members (continued)

Parameter name	Type	Description
`configuredBitRate`	integer	Configured bit rate of the channel (in bps), see `configuredBitRateReason` for why this differs from `channelBitRate`. The bitrate which TS requests from the media source.
`configuredBitRateReason`	string	One of: `aggregateBandwidth`, `flowControl`, or `notLimited`.
`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 TelePresence Server, 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. `true` if long-term reference frames are being inserted in this stream.

Table 131 `contentVideoRx` 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).
`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).

Table 131 contentVideoRx stream struct members (continued)

Parameter name	Type	Description
`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 TelePresence Server 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 132 `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).
`configuredBitRate`	integer	Configured bit rate of the channel (in bps), see `configuredBitRateReason` for why this differs from `channelBitRate`.
`configuredBitRateReason`	string	One of: `aggregateBandwidth`, `flowControl`, or `notLimited`.
`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 TelePresence Server, although a lower framerate may be sent.
`fastUpdateRequestsReceived`	integer	Number of fast update requests received.

Table 132 contentVideoTx stream struct members (continued)

Parameter name	Type	Description
`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. `true` 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 133 `flex.participant.requestPreview` inputs

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`receiverURI`	string (255)	**Required**. Fully-qualified `http` or `https` URI (for example, http://example.com:5050/RPC2 or https://example.com:5050/RPC2) 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	**Required**. Each member of the array is a Table 134 Stream struct inputs, page 97 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).
`sourceIdentifier`	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 134 Stream struct inputs

Parameter name	Type	Description
`streamIdentifier`	string	**Required**. Video stream to preview. One of `rxMainVideo`, `txMainVideo`, or `extendedVideo`. In the case of `extendedVideo`, the choice of `incoming` or `outgoing` is decided by the TelePresence Server depending on what is currently active, and this is returned in the response.

Table 134 Stream struct inputs (continued)

Parameter name	Type	Description
`position`	integer (>= 0 and <= {number of screens} - 1)	This parameter must always be supplied unless the `streamIdentifier` is `extendedVideo`. 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 `position`= 2.
`maxWidth`	integer (>= 88)	Maximum width of generated preview. Useful range 88-176 (pixels). Setting `maxWidth` > 176 will not return a wider image. Default: 88.
`maxHeight`	integer (>= 72)	Maximum height of generated preview. Useful range 72-144 (pixels). Setting `maxHeight` > 144 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 * 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 135 `participantPreviewResponse` data

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`sourceIdentifier`	string (255)	Source identifier provided in the original request (or Port A MAC address if not supplied in request).
`streams`	array of structs	Each member of the array is a Table 136 Preview struct members, page 99 which will contain the base64-encoded JPEG preview if it was retrieved.

Table 136 Preview struct members

Parameter name	Type	Description
`status`	string	Either `ok`, or one of the following strings giving the reason for failing to obtain a preview of the stream: ■ `audioOnly`: endpoint is audio-only and is not capable of receiving video. ■ `noCurrentPresentation`: currently no active extended video channel (conference has no active presentation stream). ■ `contentInMain`: there is active extended video, but this endpoint is not capable of receiving it - presentation is currently being displayed in rxMainVideo stream. ■ `internalError`: unexpected error when trying to generate preview. The two content-related statuses are returned only if the requested stream is an extended video stream.
`direction`	string	Whether the preview is of an `rx` (incoming) or `tx` (outgoing) stream.
`context`	string	Whether preview is of the `main` or `extended` stream.
`position`	integer	Position of stream starting from 0, going from left to right. For example, the right screen of a T3 would be `position` = 2.
`preview`	base64	Base64 encoded JPEG binary data (only valid if status is `ok`). The image is constrained by `maxWidth` and `maxHeight`, 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 137 flex.participant.sendDTMF inputs

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`dtmf`	string (127)	**Required**. Valid DTMF, page 20 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 138 `flex.participant.sendUserMessage` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`message`	string (500)	**Required**. 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 139 `flex.participant.setImportant` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

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 140 `flex.participant.setMute` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`audioRxMute`	boolean	Whether audio from the endpoint is muted.
`videoRxMute`	boolean	Whether the main video from the endpoint is muted.
`audioTxMute`	boolean	Whether audio to the endpoint is muted.
`videoTxMute`	boolean	Whether the main video to the endpoint is muted.

flex.participant.status

Returns the status of the specified participant.

Table 141 `flex.participant.status` **inputs**

Parameter name	Type	Description
`participantID`	string (50)	**Required**. Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.

Table 142 `flex.participant.status` **returned data**

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`participantReference`	string (50)	Client reference string. See Identifiers and Client References, page 13. Returned if string length > 0.
`displayName`	string (80)	Participant display name. Returned if string length > 0.
`conferenceID`	string (50)	Identifier of the conference to which the participant is connected or is in the process of connecting. See Identifiers and Client References, page 13.
`accessLevel`	string	Access level assigned to the participant. See Table 7 Access level enumerated type, page 23. **Note:** Incoming calls are reported as `unknown` until they have been authorized (reach the lobby screen).
`participantType`	string	Indicates the participant type. See Table 143 Participant type string values, page 102
`participantMediaInfo`	struct	The Table 144 Participant media information struct members, page 103 details the media resources and credits allocated to this participant.
`important`	boolean	Whether the participant is important.
`calls`	array of structs	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. See Table 145 Participant call status struct members, page 103. 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.
`audioRxMute`	boolean	Whether audio from endpoint is muted.
`videoRxMute`	boolean	Whether main video from endpoint is muted.
`audioTxMute`	boolean	Whether audio to endpoint is muted.
`videoTxMute`	boolean	Whether main video to endpoint is muted.

Table 142 flex.participant.status returned data (continued)

Parameter name	Type	Description
`displayName`	string (80)	Participant display name. Returned if string length > 0.
`layout`	string	The display layout that is currently shown on the participant's endpoint. One of `layoutSingle`, `layoutActivepresence`, `layoutProminent`, or `layoutEqual`. 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). See Table 10 Single screen layout enumerated type, page 24 or Table 11 Multi-screen layout enumerated type, page 24, depending on the type of endpoint.

Table 143 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
SIP three screen telepresence
Legacy TIP endpoint
Legacy single screen TIP endpoint
Legacy three screen TIP endpoint
Cascade
Multistream
Unknown

Table 144 Participant media information struct members

Parameter name	Type	Description
`mainVideoTokenInfo`	struct	A Participant token information struct for the participant's main video. See Table 144 Participant media information struct members, page 103.
`extendedVideoTokenInfo`	struct	A Participant token information struct for the participant's extended video. See Table 144 Participant media information struct members, page 103.
`audioTokenInfo`	struct	A Participant token information struct for the participant's audio. See Table 144 Participant media information struct members, page 103.
`creditsConfigured`	integer (>= 0)	Number of credits configured for the participant.
`creditsFarEnd`	integer (>= 0)	Number of credits required to match far end capability. Absent if the far-end capabilites are not yet known, or if some calls are not yet established.
`creditsNearEnd`	integer (>= 0)	Number of credits required to match near end capability. Absent if the far-end capabilites are not yet known, or if some calls are not yet established.

Table 145 Participant call status struct members

Parameter name	Type	Description
`callID`	string (50)	Call identifier. See Identifiers and Client References, page 13.
`conferenceID`	string (50)	Identifier of the conference to which the call is connected or is in the process of connecting. See Identifiers and Client References, page 13.
`conferenceState`	string	State of call connection to the conference. See Table 16 Call conference state enumerated type, page 25.
`callState`	string	State of the call. See Table 17 Call state enumerated type, page 25.
`incoming`	boolean	Direction of the call: `true` indicates incoming; `false` indicates outgoing.
`protocol`	string	Call control protocol. See Table 12 Call protocol enumerated type, page 24.
`address`	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.
`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. Only returned for strings of length > 0.

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

Table 146 `flex.reservation.create` **inputs**

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

Table 147 `flex.reservation.create` **returned data**

Parameter name	Type	Description
`reservationId`	string (50)	Reservation identifier, assigned by the DSC. All subsequent queries for this reservation can use this identifier to reference the reservation. See Identifiers and Client References, page 13.
`reservationReference`	string (50)	Client reference string, if not blank. See Identifiers and Client References, page 13.

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

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

Parameter name	Type	Description
`reservationId`	string (50)	Reservation identifier, assigned by the DSC.

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

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

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

Table 150 `flex.reservation.modify` **inputs**

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

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

Table 151 `flex.reservation.status` **inputs**

Parameter name	Type	Description
`reservationID`	string (50)	**Required**.Reservation identifier, assigned by the DSC. See Identifiers and Client References, page 13.

Table 152 `flex.reservation.status` **returned data**

Parameter name	Type	Description
`reservationID`	string (50)	Reservation identifier assigned by the TelePresence Server. See Identifiers and Client References, page 13.
`reservationReference`	string (50)	Client reference.
`numParticipantsRemaining`	integer	Number of participants still capable of using the reservation.
`numCreditsRemaining`	integer	Number of credits still reserved by the reservation.

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

Table 153 `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 154 `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 155 `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 156 `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 157 `flex.resource.query` **returned data**

Parameter name	Type	Description
maxCalls	integer (>= 0)	Maximum number of active calls.

Table 157 flex.resource.query returned data (continued)

Parameter name	Type	Description
`maxCallsPerParticipant`	integer (>= 0)	Maximum number of calls supported for any one participant.
`maxParticipants`	integer (>= 0)	Maximum number of participants.
`maxParticipantsPerConference`	integer (>= 0)	Maximum number of participants supported for any one conference.
`maxConferences`	integer (>= 0)	Maximum number of conferences.
`maxMediaTokensPerChannel`	integer (> 0)	Maximum number of tokens that can be assigned to a channel.
`mediaTokensLimit`	integer (>= 0)	Maximum number of media tokens available. This varies with the number of TelePresence Servers within the cluster.
`mediaTokensAvailable`	integer (>= 0)	Number of media tokens currently available. This varies with the number of active TelePresence Servers within the cluster.
`maxMediaCredits`	integer (>= 0)	Maximum number of credits available. This varies with installed screen licenses.
`mediaTokenLevelsMainVideo`	array of structs	Each member of the array is a Table 158 videoMediaTokenLevel struct members, page 107 struct that defines mediaToken levels for main video. The `videoMediaTokenLevel` XML-RPC struct is used to describe levels associated with mediaToken values such as the supported resolution.
`mediaTokenLevelsExtendedVideo`	array of structs	Each member of the array is a Table 158 videoMediaTokenLevel struct members, page 107 struct that defines mediaToken levels for extended video.
`mediaTokenLevelsAudio`	array of structs	Each member of the array is an Table 159 audioMediaTokenLevel struct members, page 108 that defines mediaToken levels for audio. The `audioMediaTokenLevel` struct is used to describe levels associated with mediaToken values required for audio channels.
`mediaCreditTokenRanges`	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 Media Credits, page 17.
`minCallBandwidth`	integer (> 0)	Lowest value (bits per second) that will be accepted for call bandwidths.
`maxCallBandwidth`	integer (> 0)	Highest value (bits per second) that will be accepted for call bandwidths.

Table 158 `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 Deriving the required number of media tokens.

Table 158 videoMediaTokenLevel struct members (continued)

Parameter name	Type	Description
maxVideoArea	integer (> 0)	Maximum resolution supported. Multiply required video width and height to check if a resolution is supported. A resolution is supported if: **maxVideoArea >= (requiredVideoWidth * requiredVideoHeight)**.
maxMBps	integer (>= 0)	Maximum macroblocks per second.

Table 159 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 160 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).
conferenceCreditsRequired (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 **maxMediaCredits**.

Table 160 flex.resource.status returned data (continued)

Parameter name	Type	Description
`conferenceCreditsUsed` (experimental only—do not use)	integer (>= 0)	Number of credits in use by this conference.

logs.protocols.modify

This call modifies the protocol logging settings of the device. Any protocols not listed explicitly are not modified (unless `all` is used). Special value `all` can be used to set any other protocols on or off, for example:

- `all.isIncluded=True, BFCP.isIncluded=False` means "include every protocol except BFCP" regardless of previous settings
- `all.isIncluded=False, SIP.isIncluded=True` means "include only SIP" regardless of previous settings
- `SIP.isIncluded=True` means "make sure SIP is included, don't change other settings"

Table 161 `logs.protocols.modify` inputs

Parameter name		Type	Description
`loggingEnabled`		boolean	Whether the protocol logging as a whole should be enabled or not. This is independent of whether specific protocols are included.
`protocols`		array	Structure for each configurable protocol.
	`name`	string (20)	Protocol name (often an abbreviation) for example `SIP, BFCP, H.323`. The actual protocols available will vary. Unrecognized protocol names are ignored. Matching is case-insensitive. Special value all means change values for `all` unlisted protocols.
	`isIncluded`	boolean	Whether this protocol should be included in the logging.

logs.protocols.query

This query returns the current protocol logging settings of the device.

Table 162 `logs.protocols.query` returned data

Parameter name		Type	Description
`loggingEnabled`		boolean	Whether the protocol logging as a whole is enabled or not. This is independent of whether specific protocols are included.
`messagesLogged`		integer	Number of messages currently in the log, ready for download.
`protocols`		array	Structure for each configurable protocol.
	`name`	string (20)	Protocol name (often an abbreviation) for example `SIP, BCCP, H.323.`
	`isIncluded`	boolean	Whether this protocol should be included in the logging (when enabled).

logs.syslog.modify

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

Table 163 `logs.syslog.modify` **inputs**

Parameter name		Type	Description
`servers`		array	**Optional.** Structure for each syslog server to which the output should be sent. Maximum length 4. This array replaces all existing entries.
	`address`	string (255)	**Optional.** Address of syslog server
`facilityValue`		integer	**Optional.** 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 164 `logs.syslog.query` **returned data**

Parameter name		Type	Description
`servers`		array	Structure for each syslog server to which we are sending our output. Maximum length 4.
	`address`	string (255)	Address of syslog server
	`packetsSent`	integer	Number of packets sent to server
`facilityValue`		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 165 `services.modify` **inputs**

Parameter name		Type	Description
`ephemeralPortRangeMin`		integer	**Optional**. 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.
`ephemeralPortRangeMax`		integer	**Optional**. 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.

Table 165 services.modify inputs (continued)

Parameter name			Type	Description
`ports`			array	**Optional**. An array whose members are structures representing the ethernet port/protocol version combinations on the device.
	`ethernetPort`		string (10)	**Required for every entry in array**. Identifies which physical ethernet connector is being configured. One of `A` or `B`, though only 'A' is available for use on current devices.
	`protocolVersion`		string (10)	**Required for every entry in array**. Identifies the IP protocol version, which may be `IPv4` or `IPv6`.
	`services`		array	**Required for every entry in 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)	**Required for every entry in array**. The name of the service. The available services may vary by hardware or software platform and version.
		`connectionType`	string (20)	**Required for every entry in array**. The type of service. Either `tcp` or `udp`.
		`setting`	boolean	**Optional for every entry in array**. `True` if the service should be enabled, `false` if it should be disabled.
		`portNumber`	integer	**Optional for every entry in array**. Identifies the port number on which the service is made available.

services.query

Queries the network services settings of the device.

Table 166 `services.query` returned data

Parameter name			Type	Description
Returned parameters				
`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 `A` on TelePresence Server.
	`protocolVersion`		string (10)	Identifies the IP protocol version. Either `IPv4` or `IPv6`.

Table 166 services.query returned data (continued)

Parameter name			Type	Description
	`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 167 `sip.modify` inputs

Parameter name	Type	Description
Optional parameters		
`callConfiguration`	string (20)	**Optional**. Either `useTrunk` or `callDirect`.
`outboundAddress`	string (80)	**Optional**. Address of SIP trunk destination. Ignored if `callConfiguration` is `callDirect`. May not be blank if `callConfiguration` is `useTrunk`.
`outboundDomain`	string (80)	**Optional**. The domain of the trunk destination. Ignored if `callConfiguration` is `callDirect`.
`username`	string (80)	**Optional**. The username used for SIP authentication.
`password`	string (63)	**Optional**. The password used for SIP authentication.
`outboundTransport`	string (3)	**Optional**. One of `udp`, `tcp` or `tls`. `tls` can only be configured when an encryption feature key is present.
`advertiseDualIPVersions`	string (20)	**Optional**. One of `disabled` or `useANAT`
`negotiateSRTPUsingSDES`	string (20)	**Optional**. One of `TLS only` or `always`.

Table 167 sip.modify inputs (continued)

Parameter name	Type	Description
`useLocalCertificate`	boolean	**Optional**. 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 168 `sip.query` returned data

Parameter name	Type	Description
`callConfiguration`	string (20)	Either `useTrunk` or `callDirect`.
`outboundAddress`	string (80)	Address of SIP registrar or trunk destination. Ignored if `callConfiguration` is `callDirect`. May not be blank if `callConfiguration` is `useTrunk`.
`outboundDomain`	string (80)	The domain of the trunk destination. Ignored if `callConfiguration` is `callDirect`.
`username`	string (80)	The username used for SIP authentication.
`password`	string (63)	The password used for SIP authentication.
`outboundTransport`	string (3)	One of `udp`, `tcp` or `tls`.
`advertiseDualIPVersions`	string (20)	One of `disabled` or `useANAT`.
`negotiateSRTPUsingSDES`	string (20)	One of `TLS only` or `always`. Only returned when an encryption feature key is present.
`useLocalCertificate`	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 169 `system.info` returned data

Parameter name	Type	Description
`platform`	string	The TelePresence Server's platform, as it appears in **system.xml**.
`cpuModel`	string	The TelePresence Server's Hypervisor CPU model, as it appears in **system.xml**. (Cisco TelePresence Server on Virtual Machine only)
`cpuCount`	integer	The TelePresence Server's number of Virtual CPUs. (Cisco TelePresence Server on Virtual Machine only)
`operationMode`	string	One of `standalone` (locally managed), `flexible` (remotely managed), or `slave` (slave blade in a cluster).

Table 169 system.info returned data (continued)

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

Table 169 system.info returned data (continued)

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

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:

`administrator`	This user can be used to authenticate API commands except for `user` commands.
`apiOnly`	This user can be used to authenticate API commands (in particular these ones). However API authenticated by such a user cannot modify or view `administrator` level users, including the `immortal` user.
`none`	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 170 `user.create` inputs

Parameter name	Type	Description
`newUserId`	string (64)	**Required**. User ID with which the user will be able to login.
`password`	string (31)	**Required**. The password associated with the given user ID.
`accessRights`	string (32)	**Required**. One of `administrator, apiOnly, none`.
`name`	string (31)	**Optional**. 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 171 `user.destroy` inputs

Parameter name	Type	Description
`currentUserId`	string (64)	**Required**. String with which the user can currently login. The `immortal` 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 172 `user.modify` inputs

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

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 173 `user.enumerate` inputs

Parameter name		Type	Description
`enumerateID`		string (150)	**Optional**. 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.
`max`		integer	**Optional**. 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 174 `user.enumerate` **returned data**

Parameter name		Type	Description
`moreAvailable`		boolean	Whether there are users remaining after this.
`enumerateID`		string (150)	The user enumeration token to be used to continue the enumeration. This will not be returned if the enumeration is complete.
`modifiedSinceStart`		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.
`users`		array	List of the users, which are structures as follows:
	`currentUserId`	string (64)	String with which the user can login.
	`name`	string (31)	Full name for the user, or blank if none set.
	`immortal`	boolean	`True` if this user is immortal (cannot be destroyed using this API).
	`accessRights`	string (32)	Value of `administrator`, `apiOnly` or `none`.
	`passwordChangeRequired`	boolean	`True` 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>
```

```
    <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 175 System XML contents

Node name	Node contents
manufacturer	TANDBERG
model	Telepresence Server <model number> eg. *Telepresence Server 8710*
product	TS
platform	<platform> eg. *Media 310*, *8710*, or *Virtual Machine with 16 vCPUs*
productDisplayName	*Cisco TelePresence Server*. The display name values are subject to change with new software releases, so your application should not rely on them.
platformDisplayName	<platform> eg. *Media 310*, *8710*, or *Virtual Machine with 16 vCPUs*. 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. *4.1(1.22)*
buildVersion	Build number string eg. *13.3(1.22)*
hostName	Host name of the unit
ipAddress	IPv4 address
ipAddressV6	IPv6 address
macAddress	MAC address
gatekeeperUsage	**Yes**: gatekeeper usage is enabled **No**: 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	**Yes**: registrar usage is enabled **No**: registrar usage is disabled This value is always **No** in remotely managed mode.
sipRegistrarAddress	SIP registrar host name / IP address This node is always empty in remotely managed mode.

Table 175 System XML contents (continued)

Node name	Node contents
sipRegistrarDomain	SIP registrar domain This node is always empty in remotely managed mode.
sipTrunkUsage	`Yes`: trunk usage is enabled `No`: trunk usage is disabled
sipTrunkAddress	SIP trunk host name / IP address
sipTrunkDomain	SIP trunk domain
isMaster	`Yes`: this system is a master, or it is unclustered `No`: this system is a slave
clusterType	The role of this system in a cluster. May be `unclustered`, `master`, or `slave`
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 176 System XML contents

Node name	Node contents
manufacturer	Cisco
model	TelePresence Server on <platform> eg. *TelePresence Server on Media 310*
product	TS
platform	<platform> eg. *Media 310*, *8710*, or *Virtual Machine with 16 vCPUs*
productDisplayName	*Cisco TelePresence Server*. The display name values are subject to change with new software releases, so your application should not rely on them.
platformDisplayName	<platform> eg. *Media 310*, *8710*, or *Virtual Machine with 16 vCPUs*. 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. *4.1(1.22)*
buildVersion	Build number string eg. *13.3(1.22)*
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 `unclustered`, `master`, or `slave`

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 177 System XML contents

Node name	Node contents
manufacturer	Cisco
model	TelePresence Server on <platform> eg. *TelePresence Server on Virtual Machine with 16 vCPUs*
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. *0* if AVX is not supported or *1* if AVX is supported. (Cisco TelePresence Server on Virtual Machine only)
product	TS
platform	<platform> eg. *Media 310*, *8710*, or *Virtual Machine with 16 vCPUs*
productDisplayName	*Cisco TelePresence Server*. The display name values are subject to change with new software releases, so your application should not rely on them.
platformDisplayName	<platform> eg. *Media 310*, *8710*, or *Virtual Machine with 16 vCPUs*. 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. *4.1(1.22)*
buildVersion	Build number string eg. *13.3(1.22)*
hostName	Host name of the unit
ipAddress	IPv4 address
ipAddressV6	IPv6 address
macAddress	MAC address
clusterType	Always `unclustered`. (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 178 Fault codes

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

Table 178 Fault codes (continued)

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

Table 178 Fault codes (continued)

101	`missing parameter`. 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	`invalid parameter`. 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	`malformed parameter`. 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	`request too large`. The method call contains more data than the API can accept. The maximum size of the call is 32 kilobytes.
201	`operation failed`. This is a generic fault for when an operation does not succeed as required.
202	`Product needs its activation feature key`. This request requires that the product is activated.
203	`Too many asynchronous requests`. The TelePresence Server is currently dealing with the maximum number of asynchronous requests of this type. Please retry this request later.
204	`Too many invalid keys entered. Wait 5 seconds to retry`. The TelePresence Server will not currently accept more requests to add feature keys.
205	`Duplicate user id`. A user id was given, but there is already a user with the same id.
206	`HTTPS required`. Certain requests are required to be made over HTTPS when the TelePresence Server is operating in trust licensed mode.
207	`System has been shut down`. 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>
          <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 179 API version 4.4 change summary

XML-RPC Request / Topic	Parameter	Change
flex.conference.create, page 54	`audioJoinNotification` `audioLeaveNotification`	New parameters
flex.conference.modify, page 65	`audioJoinNotification` `audioLeaveNotification`	New parameters
flex.conference.query, page 69	`audioJoinNotification` `audioLeaveNotification`	New parameters
flex.conference.create, page 54	`displayAudioAvatarMode`	New parameters
flex.conference.query, page 69	`displayAudioAvatarMode`	New parameters

Table 179 API version 4.4 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
flex.conference.sendUserMessage, page 75	`message`	Increased to 500 text characters maximum
flex.participant.sendUserMessage, page 100	`message`	Increased to 500 text characters maximum
Table 30 Audio Avatar mode enumerated type, page 29	`all` `preferVideo`	New enumerated type
Participant Call Definition Struct, page 37	`remoteAddress`	String increased from 80 to 1024 characters for Cisco TelePresence Server on Virtual Machine.
Table 10 Single screen layout enumerated type, page 24	`layoutOnePlusN`	New OnePlusN layout setting for single screen participants.
Table 33 callAttributes struct members, page 32	`multistreamMode`	Default changed to `multistreamOff`.

Table 180 API version 4.3 change summary

XML-RPC Request / Topic	Parameter	Change

flex.conference.create, page 54	`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

Table 180 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
flex.conference.modify, page 65	customPINEntryFailedMessage useCustomPINEntryFailedMessage customDisconnectPlatitudeMessage useCustomDisconnectPlatitudeMessage	New parameters
	useCustomConferenceAutoDisconnectedExitMessage customConferenceAutoDisconnectedExitMessage	Removed
flex.conference.query, page 69	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
flex.conference.sendUserMessage, page 75	message	Limited to 100 text characters
flex.conference.sendUserMessage, page 75	position	Removed
flex.participant.sendUserMessage, page 100	message	Limited to 100 text characters
flex.participant.sendUserMessage, page 100	position	Removed
flex.conference.status, page 76	conferenceCreditsAllocated conferenceCreditsRequired conferenceCreditsUsed	New parameters (experimental only—do not use)

Table 180 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change

flex.resource.status, page 108	`conferenceCreditsAllocated`
	`conferenceCreditsRequired`
	`conferenceCreditsUsed`
		New parameters (experimental only—do not use)

flex.conference.enumerate, page 62	`conferenceCreditsAllocated`
	`conferenceCreditsRequired`
	`conferenceCreditsUsed`
		New parameters (experimental only—do not use)

Table 28 Participant role enumerated type (experimental only—do not use), page 28	`any`
	`accessLevelChair`
	`accessLevelGuest`
	`cascade`
		New enumerated type (experimental only—do not use)

Table 29 Resource optimization mode enumerated type, page 29	`byParticipant` `byConference` (experimental only—do not use)	New enumerated type

Enumerated Reservation Information Struct (experimental only—do not use), page 38	`reservationID`
	`reservationReference`
	`conferenceID`
	`participantRole`
	`numParticipants`
	`maxQuality`
	`totalReservedCredits`
	`numParticipantsRemaining`
	`numCreditsRemaining`
		New struct (experimental only—do not use)

flex.reservation.create (experimental only—do not use), page 104	`reservationReference`
	`conferenceID`
	`participantRole`
	`numParticipants`
	`maxQuality`
	`reservationId`
	`reservationReference`
		New command and parameters (experimental only—do not use)

flex.reservation.query (experimental only—do not use), page 104	`reservationId`
	`reservationReference`
	`conferenceID`
	`participantRole`
	`numParticipants`
	`maxQuality`
	`totalReservedCredits`
		New command and parameters (experimental only—do not use)

flex.reservation.modify (experimental only—do not use), page 105	`reservationId`
	`reservationReference`
	`numParticipants`
	`maxQuality`
		New command and parameters (experimental only—do not use)

Table 180 API version 4.3 change summary (continued)

XML-RPC Request / Topic	Parameter	Change

flex.reservation.status (experimental only—do not use), page 105	`reservationID`
	`reservationReference`
	`numParticipantsRemaining`
	`numCreditsRemaining`
		New command and parameters (experimental only—do not use)

flex.reservation.destroy (experimental only—do not use), page 1	`reservationId`	New command and parameters (experimental only—do not use)

flex.reservation.enumerate (experimental only—do not use), page 105	`cookie`
	`max`
	`conferenceID`
	`moreAvailable`
	`reservations`
		New command and parameters (experimental only—do not use)

flex.reservation.deletions.enumerate (experimental only—do not use), page 106	`cookie`
	`max`
	`conferenceID`
	`moreAvailable`
	`IDs`
		New command and parameters (experimental only—do not use)

device.qos.modify, page 47	`ipv4Audio`, `ipv4Video`, `ipv6Audio`, `ipv6Video`	New command introduced in 4.3—documentation omission.
device.qos.query, page 48	`ipv4Audio`, `ipv4Video`, `ipv6Audio`, `ipv6Video`	New command introduced in 4.3—documentation omission.
logs.protocols.modify, page 109	`loggingEnabled`, `protocols`	New command introduced in 4.3—documentation omission.
logs.protocols.query, page 109	`loggingEnabled`, `messagesLogged`, `protocols`	New command introduced in 4.3—documentation omission.

Table 181 API version 4.2 change summary

XML-RPC Request / Topic	Parameter	Change
flex.licenseMode.modify, page 77	`newLicenseMode`, `version`, `trustSecret`	New command
flex.licenseMode.verify, page 78	None	New command
system.info, page 113	`licenseMode`: new options `flexible`, `trust`	Updated parameter
user.enumerate, page 116	`passwordChangeRequired`	New parameter
Table 33 callAttributes struct members, page 32	`displayShowEndpointNames`	Corrected default to `True`

Table 182 API version 4.1.2 change summary

XML-RPC Request / Topic	Parameter	Change
device.network.modify, page 44	`portA`, `dns`, `ipv4Enabled`, `dhcpv4`, `ipv4Address`, `ipv4SubnetMask`, `defaultipv4Gateway`, `ipv6Address`, `ipv6Conf`, `ipv6PrefixLength`, `defaultIpv6Gateway`, `ethernetAutomatic`, `speed`, `fullDuplex`, `dnsConfiguration`, `hostName`, `nameServer`, `nameServerSecondary`, `domainName`	New command
device.time.modify, page 50	`currentTime`, `ntpEnabled`, `utcOffsetHours`, `utcOffsetMinutes`, `ntpHost`	New command
device.time.query, page 50	`currentTime`, `ntpEnabled`, `utcOffsetHours`, `utcOffsetMinutes`, `ntpHost`	New command
logs.syslog.modify, page 110	`servers`, `facilityValue`	New command
logs.syslog.query, page 110	`servers`, `facilityValue`	New command
services.modify, page 110	`ephemeralPortRangeMin`, `ephemeralPortRangeMax`, `ports`, `ethernetPort`, `protocolVersion`, `services`, `serviceName`, `connectionType`, `setting`, `portNumber`	New command
services.query, page 111	`ports`, `ethernetPort`, `protocolVersion`, `services`, `serviceName`, `connectionType`, `setting`, `portNumber`, `enabled`, `ephemeralPortRangeMin`, `ephemeralPortRangeMax`	New command
sip.modify, page 112	`callConfiguration`, `outboundAddress`, `outboundDomain`, `username`, `password`, `outboundTransport`, `advertiseDualIPVersions`, `negotiateSRTPUsingSDES`, `useLocalCertificate`	New command

Table 182 API version 4.1.2 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
sip.query, page 113	`callConfiguration`, `outboundAddress`, `outboundDomain`, `username`, `password`, `outboundTransport`, `advertiseDualIPVersions`, `negotiateSRTPUsingSDES`, `useLocalCertificate`	New command
user.create, page 115	`newUserId`, `password`, `accessRights`, `name`	New command
user.destroy, page 115	`currentUserId`	New command
user.enumerate, page 116	`enumerateID`, `max`, `moreAvailable`, `modifiedSinceStart`, `users`, `currentUserId`, `name`, `immortal`, `accessRights`	New command
user.modify, page 116	`currentUserId`, `newUserId`, `name`, `password`, `accessRights`	New command

Table 183 API version 4.1 change summary

XML-RPC Request / Topic	Parameter	Change
Table 33 callAttributes struct members, page 32	`packetLossThreshold`	Modified default from 0 to 5
Table 33 callAttributes struct members, page 32	`accessLevel`	New value `unknown` added
flex.participant.query, page 89	`callAttributes`	Add note regarding `unknown` status
flex.participant.enumerate, page 85	`accessLevel`	Add note regarding `unknown` status
flex.participant.advanced.enumerate, page 78	`accessLevel`	Add note regarding `unknown` status
flex.participant.status, page 101	`accessLevel`	Add note regarding `unknown` status
Table 16 Call conference state enumerated type, page 25	`disconnecting`	Added
Table 17 Call state enumerated type, page 25	`callStateFailed`	New status added
Table 21 Multistream mode enumerated type, page 26		New function and parameters
Table 22 Participant connection state enumerated type, page 27	`disconnecting neverConnected deferred partiallyFailed audioReceiverFailed`	Add new parameters to Participant connection state enumerated type table
Table 33 callAttributes struct members, page 32	`indicateAudioOnlyParticipants`	Added

Table 183 API version 4.1 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
Table 33 callAttributes struct members, page 32	`displayForceDefaultLayout`	Added Note
Table 33 callAttributes struct members, page 32	`iXEnabled`	Modified default to True
Table 33 callAttributes struct members, page 32	`displayLayoutSwitchingMode`	Added Note
Table 33 callAttributes struct members, page 32	`multistreamMode`	Added
Table 33 callAttributes struct members, page 32	`displayHighlightActiveSpeaker`	Parameter removed
Table 36 Outgoing participant call definition struct members, page 38	`toOverride`	New Outgoing parameter added
Participant PIN Definition, page 38		New set of parameters for incoming calls

flex.conference.create, page 54	`useCustomOptionalPINEntryMessage`
	`customOptionalPINEntryMessage`
	`exitScreen`
	`useCustomConferenceAutoDisconnectedExitMessage`
	`customConferenceAutoDisconnectedExitMessage`
	`useCustomConferenceEndedExitMessage`
	`customConferenceEndedExitMessage`
	`useCustomParticipantDisconnectedExitMessage`
	`customParticipantDisconnectedExitMessage`
		Added

flex.conference.modify, page 65	`useCustomOptionalPINEntryMessage`
	`customOptionalPINEntryMessage`
	`exitScreen`
	`useCustomConferenceAutoDisconnectedExitMessage`
	`customConferenceAutoDisconnectedExitMessage`
	`useCustomConferenceEndedExitMessage`
	`customConferenceEndedExitMessage`
	`useCustomParticipantDisconnectedExitMessage`
	`customParticipantDisconnectedExitMessage`
		Added

flex.conference.destroy, page 62	`allowExitScreen`	Added

Table 183 API version 4.1 change summary (continued)

XML-RPC Request / Topic	Parameter	Change

flex.conference.query, page 69	`useCustomOptionalPINEntryMessage`
	`customOptionalPINEntryMessage`
	`exitScreen`
	`useCustomConferenceAutoDisconnectedExitMessage`
	`customConferenceAutoDisconnectedExitMessage`
	`useCustomConferenceEndedExitMessage`
	`customConferenceEndedExitMessage`
	`useCustomParticipantDisconnectedExitMessage`
	`customParticipantDisconnectedExitMessage`
		Added

Table 82 Conference information struct, page 63	`terminating`	Added
flex.conference.sendUserMessage, page 75	`position`	Modified default from 5 to 2
flex.participant.create, page 81	PIN	Deprecated
flex.participant.create, page 81	PINs	Added
flex.participant.query, page 89	PIN	Deprecated
flex.participant.query, page 89	PINs	Added
Table 143 Participant type string values, page 102		Add new table of string values
flex.participant.destroy, page 84	`allowExitScreen`	Added
Table 108 flex.participant.deletions.enumerate returned data, page 84	`sipReasonHeader`	Added
flex.participant.sendUserMessage, page 100	`position`	Modified default from 5 to 2
Table 129 videoRx stream struct members, page 93	`streamType`	Added
Table 129 videoRx stream struct members, page 93	`height`, `width`, `jitter`, `framesReceived`, `frameErrors`, `frameRate`	Modified
Table 130 videoTx stream struct members, page 94	`streamType`	Added
Table 130 videoTx stream struct members, page 94	`height`, `width`, `configuredBitRate`, `frameRate`	Modified
Table 131 contentVideoRx stream struct members, page 95	`streamType`, and marked as always `Transcoded`	Added
Table 132 contentVideoTx stream struct members, page 96	`streamType`, and marked as always `Transcoded`	Added
system.info, page 113	`depHash`	Added

Table 184 API version 4.0(2.8) change summary

XML-RPC Request / Topic	Parameter	Change
callHome.configure, page 40	`mode`, `automatic`	New command
callHome.query, page 41	`mode`, `automatic`	New command

system.info [p.1]	`cpuModel`
	`cpuCount`
	`cpuAvx`
		Added

Table 185 API version 4.0 change summary

XML-RPC Request / Topic	Parameter	Change
device.feature.add, page 43	`key`	New command
device.feature.remove, page 43	`key`	New command
device.query, page 48	`mediaResourceRestarts`, `key`, `expiry`	Added
device.query, page 48	`currentTime`, `restartTime`, `uptime`	Documentation corrected
Fault Codes, page 121	61, 204	New fault codes
system.info, page 113	`softwareVersion`	Added
Enumerated Types, page 22	▪ Table 18 Cascade roles enumerated type, page 25 ▪ Table 19 Optimization profiles enumerated type, page 26 ▪ Enumerated Types, page 22 ▪ Table 22 Participant connection state enumerated type, page 27 ▪ Table 23 Encryption status enumerated type, page 27 ▪ Enumerated Types, page 22	Added
Call Attributes Struct, page 31	`recordingDeviceIndicateOnly`, `displayLayoutSwitchingMode`, `indicateMuting`, `allowStarSixMuting`	Added
Call Attributes Struct, page 31	`videoRxFlowControlOnViewedSize`	When `true`, behavior modified. Flow control now only requested when received stream not viewed by other participants.
flex.conference.create, page 54, flex.conference.query, page 69, flex.conference.modify, page 65	`optimizationProfile`, `useCustomMutedCanUnmuteMessage`, `customMutedCanUnmuteMessage`, `useCustomMutedCannotUnmuteMessage`, `customMutedCannotUnmuteMessage`	Added

Table 185 API version 4.0 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
flex.conference.create, page 54, flex.conference.query, page 69, flex.conference.modify, page 65	`voiceSwitchingSensitivity`	Default modified from 50 to 60.
flex.conference.enumerate, page 62	`conferenceName`, `presenterID`, `importantID`	Added
flex.conference.status, page 76	`presenterID`	Added
flex.participant.enumerate, page 85	`displayName`, `connectionState`	Added
flex.participant.advanced.enumerate, page 78	Added `encryptionStatus`, `layout`, `mediaStatus`, `rxBandwidth`, `txBandwidth`, and `protocol`	New command. Alternative to `flex.participant.enumerate`
flex.participant.deletions.enumerate, page 83	`conferenceID`	Added
flex.participant.create, page 81, flex.participant.query, page 89	`cascadeRole`	Added
flex.participant.status, page 101	`layout`	Added
Feedback Events, page 21	`flexParticipantAdvancedEnum`	New feedback event
system.xml on Virtual Machine, page 120		New reference topic

Table 186 API version 3.1 change summary

XML-RPC Request / Topic	Parameter	Change
system.info, page 113	`clusterType`	Added
device.query, page 48	`activatedLicenses`	Added
Enumerated Types, page 22	Audio gain modes (`gainModeDisabled`, `gainModeAutomatic`, `gainModeFixed`) Control levels (`controlNone`, `controlLocal`, `controlConference`)	Added
Call Attributes Struct, page 31	`audioReceiveGainMode`, `deferConnect`, `alwaysReconnect`, `displayForceDefaultLayout`, `iXEnabled`	Added
Call Attributes Struct, page 31	`audioReceiveGain`	Modified. Previously, `audioReceiveGain` was always applied. In 3.1, `audioReceiveGain` is ignored unless the `audioReceiveGainMode` is `gainModeFixed`.
Call Attributes Struct, page 31	`maxTransmitPacketSize`	Description modified.
Participant Call Definition Struct, page 37	Default values	Documented

Table 186 API version 3.1 change summary (continued)

XML-RPC Request / Topic	Parameter	Change
Fault Codes, page 121 reference topic		Documented
flex.conference.create, page 54, flex.conference.modify, page 65, flex.conference.query, page 69	`conferenceDescription`, `chairControlLevel`, `guestControlLevel`	Added
flex.conference.create, page 54, flex.conference.modify, page 65, flex.conference.query, page 69	`welcomeMessageScreen`	Modified
flex.participant.create, page 81, flex.participant.query, page 89, flex.participant.sendDTMF, page 99	`dtmf`	Modified
flex.participant.deletions.enumerate, page 83	`extended`, `IDs`, `participantID`, `conferenceID`	Added
flex.participant.media.enumerate, page 87	`conferenceID`	Added
flex.participant.requestDiagnostics, page 90	`clearPathOverhead`, `clearPathRecovered`, `packetsLost`, `clearPathLTRF`, `clearPathLTRFRepaired`	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.

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. 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)