Cisco TelePresence TelePresence Server API

Product Programming Reference Guide

3.0

D15003

December 2012

Contents

Overview

This guide describes the APIs available in Version 3.0 of Cisco TelePresence Server:

- Part 1: Flexible operation mode. Describes the API available when the operation mode is set to **`flexible`**.
- Part 2: Standalone operation mode. Describes the API available when the operation mode is set to **`standalone`**.

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

Part 1: Flexible operation mode

Introduction

Part 1 of this guide describes the API available in flexible operation mode. For information about the API available in standalone operation mode, refer to Part 2.

Terminology

This guide uses the following conventions:

Term	Meaning
Identifier	A string of characters associated with an entity or resource.
Client reference	A string of characters associated with an entity created using the API. Client reference contents are set by the API client.
Participant	An entity connected to a conference through one or more calls.

API overview

API implementation

XML-RPC

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>20050218T10:45:00</dateTime.iso8601></value>
</member>
```

Note: For string parameters, the number shown in the Type column of the relevant reference table indicates the maximum number of characters permitted.

For more information about XML-RPC see the XML-RPC specification[1].

Transport

The device implements HTTP/1.1 as defined by RFC 2616[2]. It expects to receive HTTP communications over TCP/IP connections to port 80. The application should send HTTP POST messages to the URL defined by path **/RPC2** on the device's IP address.

The device also supports HTTPS, provided that it is running software version 2.0 or later.

By default, HTTPS is provided on TCP port 443. Optionally, you can configure the device to receive HTTP and HTTPS connections on non-standard TCP port numbers.

Consider API overhead when writing applications

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).

It is important to bear this overhead in mind when designing your application architecture and software. The Design Considerations section provides recommendations for minimizing API overhead.

Design considerations

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.

HTTP keep-alives

Your application can use HTTP keep-alives to reduce the amount of TCP traffic that results from constantly polling the device.

Implementation

Any client that supports HTTP keep-alives may include the following line in the HTTP header of an API request:

```
Connection: Keep-Alive
```

This line indicates to the device that the client supports HTTP keep-alives. The device *may* then decide that it will maintain the TCP connection after it has responded to the request.

If the device decides that it will not maintain the connection after it has responded, the device returns the following line in the HTTP header of its response:

```
Connection: close
```

Subject to the limitations mentioned below, if this line is absent from the HTTP header of the response, the device will keep the TCP connection open and the client may use the same connection for a subsequent request.

Limitations

The device will not allow a connection to be kept alive if:

- The current connection has already serviced a set number of requests.
- The current connection has already been open for a set amount of time.
- There are already more than a certain number of connections in a kept-alive state.

These restrictions are in place to limit the resources associated with kept-alive connections. If a connection is terminated for either of the first two reasons, the client will probably find that the connection is maintained (back in a keep-alive state) after the next request.

Usage considerations

The client should never assume that a connection will be maintained.

Also, even after a response that does not contain the **`Connection:close`** header, the device will close an open connection if no further requests are made by the client within one minute. If requests from the client are likely to be this far apart, there is little to be gained by using HTTP keep-alives.

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 in the reference section.
  - 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.

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>
```

Authentication

All method requests must have the authentication fields listed in the following table.

Parameter name	Type	Description
`authenticationUser`	string	User name.
`authenticationPassword`	string	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.

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.
- Multi-screen, single-call connection.
- Multi-screen, multi-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.

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.

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`** 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.

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.

Integer fields and encoding of "unlimited"

This API uses a combination of a boolean field and an integer field to encode the values for settings that can have infinity as a valid value. The naming convention for the boolean field is to append **`Unlimited`** to the associated integer field.

When these values are exchanged, either the boolean or the integer field is required but not both. The TelePresence Server adheres strictly to this rule when returning values.

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`**.

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

When supplying values:

- Only one field is required.
- The boolean value is implicitly set to **`false`** if only the integer value is supplied.
- If the boolean value is **`true`**, the integer value must not be supplied.

Integer value ranges

Integer values are restricted to the following ranges:

- The maximum value (unless otherwise specified) for integer fields is 2147483647.
- The minimum value (unless otherwise specified) for integer fields is -2147483648.
- The two limits above apply to all integer fields that do not have any explicitly specified limits; that is, the range of values is from -2,147,483,648 to 2,147,483,647.

Media credits

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

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

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

Media credits	Media tokens
48	up to 48
630	up to 630
840	up to 840
1260	up to 1260
2520	up to 2520
3780	up to 3780
5040	up to 5040
7560	up to 7560
10080	up 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.

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 Example use of enumeration to optimize token usage.

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

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

Feedback events and enumeration

Feedback events are generated to aid incremental enumeration of conferences and participants. See Feedback events for more information.

For conferences:

- When a conference is created, modified or its state changes, the **`flexConferenceEnum`** event is generated. Invoke **`flex.conference.enumerate`** 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`**.
- When a conference is destroyed, the **`flexConferenceDeletionsEnum`** event is generated. Invoke **`flex.conference.deletions.enumerate`** 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`** 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`** method.
- When a participant is created or its media resource state changes, the **`flexParticipantMediaEnum`** event is generated. Invoke **`flex.participant.media.enumerate`** 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`** method.
- When a participant is destroyed, the **`flexParticipantDeletionsEnum`** event is generated. Invoke **`flex.participant.deletions.enumerate`** to identify which participants have been destroyed.

DTMF

The set of valid characters for DTMF is:

- *#0123456789ABCD,

The comma ',' is used for spacing. Each comma denotes a two-second delay.

DTMF strings that contain characters other than these are invalid parameters.

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.

- Use **`feedbackReceiver.configure`** to register a receiver to listen for one or more feedback events.
- Use **`feedbackReceiver.query`** to return a list of receivers that are configured on the device.
- Use **`feedbackReceiver.reconfigure`** to change the configuration of an existing feedback receiver.
- Use **`feedbackReceiver.remove`** to remove an existing feedback receiver.
- Use **`feedbackReceiver.status`** to display the status of a specific feedback receiver.

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 otherwise will be the device's MAC address.
- **`events`** is an array of strings that contain the names of the feedback events that have occurred.

Feedback events

Entries in the events array can be one or more of the following

Name	Description
`restart`	TelePresence Server has restarted or booted.
`configureAck`	A feedback receiver has been added, reconfigured or removed.
`deviceStatusChanged`	Shutdown status has changed. Invoke **`device.query`** for more information.
`cdrAdded`	A call record has been added to the log.
`flexConferenceEnum`	The state of one or more conferences has changed. Invoke the **`flex.conference.enumerate`** method to get the changes.
`flexConferenceDeletionsEnum`	One or more conferences have been destroyed. Invoke the **`flex.conference.deletions.enumerate`** method to get the identifiers of conferences destroyed.
`flexParticipantEnum`	Participants have been created or their state has changed. Invoke the **`flex.participant.enumerate`** method to get the changes.
`flexParticipantMediaEnum`	Participants have been created or their media resources state has changed. Invoke the **`flex.participant.media.enumerate`** method to get the changes.
`flexParticipantDeletionsEnum`	Participants have been destroyed. Invoke the **`flex.participant.deletions.enumerate`** method to get the identifiers of participants destroyed.
`flexResourceConfiguration`	The resource configuration has changed. Media blades have been added or removed. Invoke the **`flex.resource.query`** method to retrieve the new configuration.
`flexResourceStatus`	Resource usage has changed: calls, participants, conferences, media tokens, and media credits. Invoke the **`flex.resource.status`** method to get the changes.
`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.

Name	Description
`receiverModified`	The feedback receiver receiving this event has been modified.
`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.

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`**; only if there is no response should the server be considered no longer alive.

Data structures and types

Enumerated types

This API introduces the concept of 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 field is described as belonging to a particular enumeration:

- For all input strings not belonging to that list, an invalid parameter fault will be generated.
- Only strings belonging to the list will be returned by TelePresence Server.
- The maximum length of the string returned is the length of the longest string in the associated list.

Access level

This enumerated type describes the access levels that can be granted to a participant.

Name	Description
`chair`	The participant is granted chair access to the conference.
`guest`	The participant is granted guest access to the conference.

Motion sharpness

This enumerated type describes the motion sharpness settings for a participant.

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

Picture aspect ratio

This enumerated type describes the aspect ratio settings for a participant.

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

Single-screen layout

This enumerated type describes the layout settings for single-screen participants.

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.

Multi-screen layout

This enumerated type describes the layout settings for multi-screen participants.

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

Protocol

This enumerated type describes the call control protocol.

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

Video format

This enumerated type describes the participant video format.

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.

Participant encryption options

This enumerated type describes the participant encryption options.

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

Video transmit size

This enumerated type describes the participant video transmission size options.

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.

Call conference state

This enumerated type describes the state of connection between a call and a conference.

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.

Call state

This enumerated type describes the state of a call.

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.

Full screen modes

This enumerated type describes the settings for full-screen viewing of single-screen participants.

Name	Description
`never`	This single-screen participant is never shown in full-screen panes.
`always`	This single-screen participant is always allowed to be shown in full-screen panes.
`dynamic`	This single-screen participant is allowed to be shown in full-screen panes if there are no multi-screen participants to show. However, when there are multi-screen participants to show, this single-screen participant will then be restricted to the smaller continuous presence panes.

Structs

The following structs are used by multiple methods in this API. Other structs that are applicable to one method alone are described under the associated method in the reference section.

- Media tokens struct
- Participant media resources struct
- Call attributes struct
- Conference URI details struct
- Participant call definition struct

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.

Input parameters

Required inputs

Parameter name	Type	Description
`total`	integer (>= 0)	Maximum total media token usage permitted across all channels.

Optional or conditional inputs

Parameter name	Type	Description
`maxPerChannel`	integer (>= 0, <= **`total`**)	Maximum media resource usage permitted for a channel. Default: see **`maxPerChannelUnlimited`**.
`maxPerChannelUnlimited`	boolean	Whether an unlimited number of resources can be used for a channel. See Integer fields and encoding of "unlimited". Default: **`true`**.

How media token parameters are interpreted by the TelePresence Server

In general, the following rules apply:

- 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`** method and is the value of the **`maxMediaTokensPerChannel`** field.

How media token parameters are returned by the TelePresence Server

The following rules apply to the return of media token parameters:

- 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.

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

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.

`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.

`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

Call attributes are collections of parameters that describe attributes of a call. They can be specified in the following places:

- Participant specification (see **`flex.participant.create`** method).
- Participant modification (see **`flex.participant.modify`** method).
- Conference URI specification (see **`Conference URI details`** struct).
- Conference creation (see **`flex.conference.create`** method).
- Conference modification (see **`flex.conference.modify`** method).

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

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this struct.

Parameter name	Type	Description
`accessLevel`	string	Access level associated with this participant, when connected. See Access level enumerated type. Default: **`chair`**.
`encryption`	string	Encryption setting. See Participant encryption options. Default: **`optional`**.
`autoDisconnect`	boolean	Whether this call automatically disconnects if the only calls connected to a conference have **`autoDisconnect`** set. Default: **`false`**.
`maxTransmitPacketSize`	integer (0–1522)	Limit on the maximum transmission size (MTU) of packets to prevent breakup by routers. Default: 1400.
`packetLossThreshold`	integer (0–100)	Packet losses are reported when the proportion of packets lost (percentage * 10) exceeds this threshold over a short period of time. Default: 0.
`videoRxFlowControlOnErrors`	boolean	Flow control on video errors. Default: **`true`**.
`videoRxFlowControlOnViewedSize`	boolean	Flow control based on viewed size. Default: **`true`**.
`videoTxSizeOptimization`	string	Video transmit size setting. See Video transmit size enumerated type. 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`**.

Parameter name	Type	Description
`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`**. 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	Whether the audio only icon is displayed. Default: **`true`**.
`mainVideoTxPictureAspectRatio`	string	Permissible aspect ratios for the main video channel. See Picture aspect ratio enumerated type. Default: **`onlySixteenToNine`**.
`extendedVideoTxPictureAspectRatio`	string	Permissible aspect ratios for the extended video channel. See Picture aspect ratio enumerated type. Default: **`allowAllResolutions`**.
`videoTxFormat`	string	Video format. See Video format enumerated type. Default: **`NTSC`**.
`videoTxMotionSharpness`	string	Motion sharpness setting. See Motion sharpness enumerated type. 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 Full screen modes enumerated type. 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 Single-screen layout enumerated type. Default: **`layoutActivePresence`**.
`displayDefaultLayoutMultiScreen`	string	Layout scheme used for multi-screen endpoints. See Multi-screen layout enumerated type. Default: **`layoutActivePresence`**.
`displayShowEndpointNames`	boolean	Whether endpoint names are shown as panel labels. Default: **`false`**.

Parameter name	Type	Description
displayHighlightActiveSpeaker	boolean	Whether the active speaker is highlighted with a distinctive border. Default: **true**.
audioReceiveGain	integer (-12000 to +12000)	Gain for received audio, measured in millidecibels. Default: 0.
audioTransmitGain	integer (-12000 to +12000)	Gain for transmitted audio, measured in millidecibels. Default: 0.
forceTIP	boolean	Whether the use of TIP (Telepresence Interoperability Protocol) is enforced, even if the endpoint does not indicate that it is TIP capable. Default: **false**.
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. Default: **false**.
recordingDevice	boolean	Whether this call is treated as a recording device by muting received video and displaying a red dot on other endpoints. Default: **false**.

How call attributes are derived

Call attributes for a participant are created by overlaying two instances of call attributes:

Either

- Call attributes defined for a participant
- Conference default call attributes

Or

- Call attributes defined for a conference URI
- Conference default call attributes

The conference default call attributes are used to define default values for call attribute member fields not explicitly defined for the conference URIs or participants.

If no **callAttributes** members are defined for the conference URIs and participants, all participants connected to the conference will have the same call attribute values, which are the conference default call attribute values.

The values of all **`callAttributes`** fields are set when a participant is instantiated. This means that subsequent changes to conference default call attributes or conference URI call attributes have no effect on existing participants. And queries on participant call attributes will always return values for all members.

For members of the **`callAttributes`** struct whose values have not been set explicitly, the values will be the default values listed in the previous table.

Conference URI details struct

The conference URI details struct defines conference URIs, associated access levels, and media resources.

Required inputs

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

Parameter name	Type	Description
`URI`	string (80)	String used by endpoints to connect to this conference. See Conference URI.
`callBandwidth`	integer (>= **`minCallBandwidth`** and <= **`maxCallBandwidth`**)	Connection bandwidth measured in bits per second. See **`flex.resource.query`**.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this struct.

Parameter name	Type	Description
`PIN`	string (40)	PIN for the conference at this URI and access level. Default: empty.
`callAttributes`	**`callAttributes`** struct	Attributes applied to calls connecting to a conference using the aforementioned URI. See How call attributes are derived. Default: default conference call attributes, see **`flex.conference.create`** and **`flex.conference.modify`**.
`participantMediaResources`	**`participant Media Resources`** struct	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.

Incoming participant call definition struct

Required inputs

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

Parameter name	Type	Description
`URI`	string (80)	String used to connect to this conference. See Participant conference URIs.
`callBandwidth`	integer (>= **`minCallBandwidth`** and <= **`maxCallBandwidth`**)	Connection bandwidth measured in bits per second. See **`flex.resource.query`**.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this struct.

Parameter name	Type	Description
`disconnectOnIncoming`	boolean	Whether the existing call (if one exists) is disconnected when an incoming call comes through the **`URI`**. Default: **`false`**.

Outgoing participant call definition struct

Required inputs

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

Parameter name	Type	Description
`remoteAddress`	string (80)	Address of the endpoint expressed in the form of an endpoint address or E164 number.
`protocol`	string	Call control protocol. See Protocol enumerated type.
`callBandwidth`	integer (>= **`minCallBandwidth`** and <= **`maxCallBandwidth`**)	Connection bandwidth in bits per second. See **`flex.resource.query`**.

API method reference

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

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

- Description of the method's effect
- Accepted parameters, and whether they are required or optional
- Returned parameters, and whether they are always or conditionally returned

flex.resource.query

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

Returned data

The following table lists the parameters returned by this method.

Parameter name	Type	Description
`maxCalls`	integer (>= 0)	Maximum number of active calls.
`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 **`videoMediaTokenLevel`** structs	Settings associated with mediaToken levels for main video.
`mediaTokenLevelsExtendedVideo`	array of **`videoMediaTokenLevel`** structs	Settings associated with mediaToken levels for extended video.

Parameter name	Type	Description
mediaTokenLevelsAudio	array of **audioMediaTokenLevel** structs	Settings associated with mediaToken levels for audio.
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.
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.

Video media token level struct

The **videoMediaTokenLevel** XML RPC struct is used to describe levels associated with mediaToken values such as the supported resolution.

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.
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.

Deriving the required number of media tokens

Clients 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.

Audio media token level struct

The **audioMediaTokenLevel** XML RPC struct is used to describe levels associated with mediaToken values required for audio channels.

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 method takes no input parameters.

Returned data

The following table lists the parameters that are returned by this method.

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.

flex.conference.create

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

Input parameters

Required inputs

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

Parameter name	Type	Description
`participantMediaResources`	**`participantMediaResources`** struct	Conference default participant media resource configuration. These settings are overridden by those defined for the conference URI or the participant.

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

- **`participantMediaResources.mediaTokensMainVideo.total`**
- **`participantMediaResources.mediaTokensExtendedVideo.total`**

- **`participantMediaResources.mediaTokensAudio.total`**
- **`participantMediaResources.numMediaCredits`**

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
`conferenceReference`	string (50)	Conference client reference string. See Identifiers and client references. Default: empty.
`conferenceName`	string (80)	Human readable label for the conference. Default: empty.
`URIs`	array of **`Conference URI details`** structs	Conference URIs and associated access levels and media tokens. A maximum of two conference URIs are supported. Default: empty array.
`conferenceMediaTokens`	integer (>= 0)	Maximum number of media tokens that can be used for the conference. Default: see **`conferenceMediaTokensUnlimited`**.
`conferenceMediaTokensUnlimited`	boolean	Whether no limit is defined for the number of media tokens that can be used for the conference. See Integer fields and encoding of "unlimited". Default: **`true`**.
`conferenceMediaCredits`	integer (>= 0)	Maximum number of media credits that can be used for the conference. Default: see **`conferenceMediaCreditsUnlimited`**.
`conferenceMediaCreditsUnlimited`	boolean	Whether no limit is defined for the number of media credits that can be used for the conference. See Integer fields and encoding of "unlimited". 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. Default: see **`durationUnlimited`**.

Parameter name	Type	Description
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. See Integer fields and encoding of "unlimited". Default: **true**.
billingCode	string (80)	Billing code string. Default: empty.
callAttributes	**callAttributes** struct	Conference default call attributes. See How call attributes are derived. Default: see Call attributes struct.
maxParticipants	integer (>= 0)	Maximum number of participants that can connect to this conference. Default: see **maxParticipantsUnlimited**.
maxParticipantsUnlimited	boolean	Whether an unlimited number of participants are allowed to connect to this conference. See Integer fields and encoding of "unlimited". Default: **true**.
voiceSwitchingSensitivity	integer (0–100)	Voice switching sensitivity. Default: 50.
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 for this conference. If this message is empty, the conference name is displayed as the welcome screen message. Default: empty.
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.
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.
useCustomWaitingForChairMessage	boolean	Whether a custom message is displayed when waiting for a chair to join the conference. Default: **false**.
customWaitingForChairMessage	string (500)	Custom message displayed when waiting for a chair to join the conference. Only used if **useCustomWaitingForChairMessage** is **true**. Default: empty.

Parameter name	Type	Description
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.
use CustomConferenceEndingMessage	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**.

The following pairs of parameters must be specified in compliance with the requirements stated in Integer fields and encoding of "unlimited":

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

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. All subsequent invocations of methods to control or query this conference must use this identifier to reference it. See Identifiers and client references.

Conditionally returned

The following table lists the parameters that are conditionally returned by this method.

Parameter name	Type	Description
conferenceReference	string (50)	Conference client reference string. Returned if string length > 0. See Identifiers and client references.

flex.conference.modify

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

Input parameters

Required inputs

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

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.

Optional or conditional inputs

The following table lists the optional and conditional inputs taken by this method.

Parameter name	Type	Description
`conferenceReference`	string (50)	Conference client reference string. See Identifiers and client references.
`conferenceName`	string (80)	Human readable label for the conference.
`URIs`	array of **`Conference URI detail`** structs	Conference URIs, associated access levels, and media resources. Replaces all earlier conference URI entries. For URIs specified, all fields are set; unspecified optional fields are set to default values. See Conference URI details struct. If this array is empty, the previously defined conference URIs (if any) are deleted.
`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.

Parameter name	Type	Description
`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`	**`callAttributes`** struct	Conference default call attributes. See How call attributes are derived.
`participantMediaResources`	**`participant Media Resources`** struct	Conference default participant media resource configuration. These settings are 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 Integer fields and encoding of "unlimited".
`voiceSwitchingSensitivity`	integer (0–100)	Voice switching sensitivity.
`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.
`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`**.

Parameter name	Type	Description
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**.
use CustomOnlyVideoParticipantMessage	boolean	Whether a custom message is displayed when a participant is the only (active) video participant.
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 meta data.
unlockWithLastCall	boolean	Whether the conference is unlocked when the participant leaves the conference.

The following pairs of parameters must be specified in compliance with the requirements stated in Integer fields and encoding of "unlimited":

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

Only those parameters specified are changed. Other settings are left as they are unless implicitly affected by other settings that have changed. For example, if **duration** is specified, **durationUnlimited** is implicitly set to **false**.

flex.conference.query

Returns the parameters of a conference.

Input parameters

Required inputs

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

Parameter name	Type	Description
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.
URIs	array of **Conference URI details** structs	Conference URIs and associated access levels and media tokens.
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	**call Attributes** struct	Conference default call attributes. See How call attributes are derived.
voiceSwitchingSensitivity	integer (0–100)	Voice switching sensitivity.
participantMediaResources	**participant Media Resources** struct	Conference default participant media resource configuration. These settings are 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.

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.
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**.
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.

Conditionally returned

The following table lists the parameters that are conditionally returned by this method.

Parameter name	Type	Description
conferenceReference	string (50)	Conference client reference string. See Identifiers and client references.

Parameter name	Type	Description
conferenceName	string (80)	Human readable label for the conference.
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.
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. See **durationUnlimited**.
durationUnlimited	boolean	Whether an unlimited duration is assigned for the conference.
billingCode	string (80)	Billing code string.
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 Integer fields and encoding of "unlimited".

The following pairs of parameters must be specified in compliance with the requirements stated in Integer fields and encoding of "unlimited":

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

flex.conference.destroy

Destroys the specified conference. No parameters are returned.

Input parameters

Required inputs

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

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.

flex.conference.status

Returns the status of the specified conference.

Input parameters

Required inputs

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

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.
`locked`	boolean	Whether the conference is locked.
`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.

Conditionally returned

The following table lists the parameters that are conditionally returned by this method.

Parameter name	Type	Description
conferenceReference	string (50)	Conference client reference string. Returned if string length > 0. See Identifiers and client references.
endTime	integer	Amount of time, measured in seconds, until the conference is due to end. Only returned if conference duration is limited.

flex.conference.getMetadata

Returns metadata associated with the specified conference.

Input parameters

Required inputs

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

Parameter name	Type	Description
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
metadata	base64 (<= 512 bytes)	Client meta data.

If a conference does not have metadata, 0 length meta data is returned.

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 method. See Enumeration.

Input parameters

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

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.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`moreAvailable`	boolean	Whether there are more conferences to be enumerated.
`cookie`	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
`conferences`	array of **`enumerated conference information`** structs	Array of conference information structs. This array can be empty.

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.

Enumerated conference information struct

Returned data

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

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.
`locked`	boolean	Whether the conference is locked.
`active`	boolean	Whether the conference has started.
`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.

Parameter name	Type	Description
`callTokensAllocatedMainVideo`	integer (>= 0)	Number of tokens allocated for main video summed over all participants connected to the conference.
`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.

Conditionally returned

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

Parameter name	Type	Description
`conferenceReference`	string (50)	Conference client reference string. Returned if string length > 0. See Identifiers and client references.

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` method.

flex.conference.deletions.enumerate

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

Input parameters

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

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.

Returned data

The following table lists the parameters that are returned by this method.

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.

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.

Input parameters

Required inputs

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

Parameter name	Type	Description
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.
message	string (500)	Message to display.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
position	integer (1–9)	Position on the display: | 1 2 3 | | 4 5 6 | | 7 8 9 | Default: 5.
duration	integer (>= 0)	Duration in seconds for the message display on the endpoint. Default: 30 seconds.

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.

Input parameters

Required inputs

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

Parameter name	Type	Description
conferenceID	string (50)	Conference identifier assigned by the TelePresence Server. See Identifiers and client references.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
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.participant.create

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

Input parameters

Required inputs

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

Parameter name	Type	Description
`conferenceID`	string (50)	Conference identifier. See Identifiers and client references.
`calls`	array of **`participant call definition`** structs	List of call specifications for this participant; minimum of 1 and a maximum of 4. The position of a struct corresponds to a call's physical location. So for a three-call participant, position 0 is left, 1 is center, and 2 is right. For endpoints that automatically negotiate extra screens (such as a T3), it is only necessary to specify the main call at position 0; the remaining calls will be added as they are negotiated.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
`participantReference`	string (50)	Client reference string. See Identifiers and client references. Default: empty.
`PIN`	string (40)	Numeric PIN. Default: empty.
`callAttributes`	**`callAttributes`** struct	Attributes of the calls. Settings defined here override conference default call attribute settings. See How call attributes are derived. Default: conference default call attributes.
`participantMediaResources`	**`participant Media Resources`** struct	Participant media resource configuration. Settings defined here override the conference default participant media resource configuration. Default: 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.

Parameter name	Type	Description
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 (32)	Valid DTMF characters representing the **dtmf** sequence to send to the call nominated the **audioIndex**. Only sent when dialing out; not continuously for the duration of the call. Not sent for incoming calls. See DTMF. 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.

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

- The **audioIndex** and **contentIndex** values are invalid.
- The rules for participant call definition struct are not met

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. All subsequent invocations of methods to control or query this participant must use this identifier to reference it. See Identifiers and client references.

Conditionally returned

The following table lists the parameters that are conditionally returned by this method.

Parameter name	Type	Description
participantReference	string (50)	Client reference string. Returned if not empty. See Identifiers and client references.

Media resources (tokens and credits) for participants are reserved (see Media reservation).

flex.participant.modify

Modifies the call attributes, media resources, and display name of the specified participant. Call attributes of all member calls are changed. Media resources are distributed as described in Participant media distribution.

Input parameters

Required inputs

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

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
`displayName`	string (80)	Configured display name for the endpoint. This overrides the endpoint display name setting.
`callAttributes`	**`callAttributes`** struct	Attributes of the calls. Settings defined here override conference default call attribute settings. See How call attributes are derived.
`participantMediaResources`	**`participant Media Resources`** struct	Participant media resource configuration. Settings defined here override the conference default participant media resource configuration. Updates to **`participantMediaResources`** change all members of the struct: unspecified optional fields are set to their default values. See Participant media resources struct.

Only those parameters specified are changed.

flex.participant.query

Returns the parameters of the specified participant.

Input parameters

Required inputs

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

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
`conferenceID`	string (50)	Conference identifier. See Identifiers and client references.
`PIN`	string (40)	Numeric PIN.
`callAttributes`	`callAttributes` struct	Attributes of the calls. See How call attributes are derived.
`participantMediaResources`	`participant Media Resources` struct	Participant media resource configuration.
`calls`	array of `participant call definition` structs	List of call specifications for this participant; minimum of 1 and a maximum of 4.
`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.

Conditionally returned

The following table lists the parameters that are conditionally returned by this method.

Parameter name	Type	Description
`participantReference`	string (50)	Client reference string. See Identifiers and client references.
`displayName`	string (80)	Configured display name for the endpoint. This overrides the endpoint display name setting.
`dtmf`	string (50)	Valid DTMF characters representing the `dtmf` sequence to send to the call nominated the `audioIndex`. Not sent for incoming calls. See DTMF.
`callerName`	string (80)	Calling name seen by the endpoint. Not used for incoming calls.
`callerAddress`	string (80)	Calling address seen by the endpoint. Not used for incoming calls.

flex.participant.destroy

Destroys the specified participant. Any existing calls are destroyed.

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

flex.participant.status

Returns the status of the specified participant.

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
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.
accessLevel	string	Access level assigned to the participant. See Access level enumerated type.
participantMediaInfo	**participant media information** struct	Media resource information for the participant.
important	boolean	Whether the participant is important.
calls	array of **participant call status**	Call status for calls associated with each participant. If a call is absent, the array member is empty.

Parameter name	Type	Description
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.

Conditionally returned

The following table lists the parameters that are conditionally returned by this method.

Parameter name	Type	Description
participantReference	string (50)	Client reference string. Returned if not empty. See Identifiers and client references.
displayName	string (80)	Participant display name. Returned if string length > 0.

Participant media information struct

Returned data

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

Parameter name	Type	Description
mainVideoTokenInfo	**participant token information** struct	Participant token information for main video.
extendedVideoTokenInfo	**participant token information** struct	Participant token information for extended video.
audioTokenInfo	**participant token information** struct	Participant token information for audio.
creditsConfigured	integer (>= 0)	Number of credits configured for the participant.

Conditionally returned

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

Parameter name	Type	Description
creditsFarEnd	integer (>= 0)	Number of credits required to match far end capability. Absent if all participant calls have not been established.
creditsNearEnd	integer (>= 0)	Number of credits required to match near end capability. Absent if all participant calls have not been established.

Participant call status

An entry for participant call status may be empty if no call exists. If a call exists, the following tables apply.

Returned data

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

Parameter name	Type	Description
callID	string (50)	Call identifier. See Identifiers and client references.
conferenceState	string	State of call connection to the conference. See Call conference state.
callState	string	State of the call. See Call state.
incoming	boolean	Direction of the call: **true** indicates incoming; **false** indicates outgoing.
protocol	string	Call control protocol. See Protocol enumerated type.
address	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.

Conditionally returned

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

Parameter name	Type	Description
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.participant.call.disconnect

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

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
position	integer (>= 0)	Relative position of the endpoint in the group. 0 represents the leftmost screen from a viewing position.

This method 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.

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

flex.participant.enumerate

Enumerates participants. See Enumeration.

Input parameters

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

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. Default: none.

Returned data

The following table lists the parameters that are returned by this method.

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.
participants	array	Array of participant information structs. See Enumerated participant information struct. This array may be empty.

Enumerated participant information struct

Returned data

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

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
`conferenceID`	string (50)	Conference to which the participant is connected. See Identifiers and client references.
`accessLevel`	string	Access level granted to the participant. See Access level enumerated type.
`calls`	Array of **`enumerated call information`** struct	Call identifiers for calls associated with each participant. If a call is absent, the array member is empty. Up to 50 characters long. See Identifiers and client references.
`addresses`	array of **`enumerated address information`** struct	The address/URI information for the calls associated with each participant. The position in the array matches that of the associated call in the calls array.

Conditionally returned

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

Parameter name	Type	Description
`participantReference`	string (50)	Client reference string. See Identifiers and client references. Returned if not empty.

Enumerated call information struct

Returned data

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

Parameter name	Type	Description
`callID`	string (50)	Call identifier assigned by the TelePresence Server. See Identifiers and client references.
`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.

Enumerated address information struct

Conditionally returned

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

Parameter name	Type	Description
URI	string (80)	URI used or to be used by the endpoint to connect to the conference. Only returned for strings of length > 0. Incoming calls only.
remoteAddress	string (80)	Remote address specified in **flex.participant.create** to call out to the endpoint. Only returned for strings of length > 0. Outgoing calls only.

Only one of **URI** or **remoteAddress** will be returned, depending on whether the call is incoming or outgoing.

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.

Input parameters

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

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. Default: none.

Returned data

The following table lists the parameters that are returned by this method.

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 **Enumeration participant media information** struct	Array of participant media resource settings. This array may be empty.

Enum participant media info struct

Returned data

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

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
`mainVideoTokenInfo`	**`Participant token information`** struct	Participant token information for main video.
`extendedVideoTokenInfo`	**`Participant token information`** struct	Participant token information for extended video.
`audioTokenInfo`	**`Participant token information`** struct	Participant token information for audio.
`creditsConfigured`	integer (>= 0)	Number of credits configured for the participant.

Conditionally returned

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

Parameter name	Type	Description
`participantReference`	string (50)	Client reference string. See Identifiers and client references. Returned if not empty.
`creditsFarEnd`	integer (>= 0)	Number of credits required to match far end capability. Absent if all participant calls have not been established.
`creditsNearEnd`	integer (>= 0)	Number of credits required to match near end capability. Absent if all participant calls have not been established.

Participant token information struct

Returned data

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

Parameter name	Type	Description
`maxTokensConfigured`	integer (>= 0)	Maximum number of tokens with respect to conference or participant configuration.

Conditionally returned

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

Parameter name	Type	Description
`maxTokensPerChannelConfigured`	integer (>= 0)	Maximum number of tokens per channel with respect to the conference or participant configuration. See Note.
`maxTokensPerChannelConfiguredUnlimited`	boolean	Whether there is an unlimited maximum number of tokens per channel with respect to the conference or participant configuration. See Note.
`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 for reserved resources with no active calls.
`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 for reserved resources with no active calls.
`maxTokensPerChannelNearEnd`	integer (>= 0)	Maximum number of tokens per channel, advertised by the TelePresence Server. Absent for reserved resources with no active calls.
`maxTokensNearEnd`	integer (>= 0)	Maximum number of tokens advertised by the TelePresence Server. Absent for reserved resources with no active calls.

Note

Exactly one of the parameters **`maxTokensPerChannelConfigured`** and **`maxTokensPerChannelConfiguredUnlimited`** is returned.

Example use of enumeration to optimize token usage

Token usage for conference can be *optimized* incrementally using **`flex.participant.media.enumerate`** and preserving the cookie value.

1. Start enumeration using conference ID.
2. Enumerate all participants in the conference (using the cookie) and optimize until no more participants are returned (**`moreAvailable`** = **`false`**)
3. Store the cookie.
4. Wait for conference enumeration to indicate that participants connected to the conference can be optimized.
5. Repeat from step 2.

Media credit usage can be optimized in a similar way.

To maintain a live set of media information on all participants, use **`flex.participant.deletions.enumerate`** to remove disconnected participants from the set.

flex.participant.deletions.enumerate

Enumerates participant deletions. The enumeration returns identifiers of participants that have been newly deleted.

Input parameters

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

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.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`moreAvailable`	boolean	Whether there are more participant deletions to be enumerated.
`cookie`	string (150)	Cookie that must be returned in the next invocation to continue the enumeration.
`participantIDs`	array of identifiers	Identifiers of participants that have been deleted. See Identifiers and client references.

flex.participant.requestPreview

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

Input parameters

Required inputs

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

Parameter name	Type	Description
`participantID`	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

Parameter name	Type	Description
`receiverURI`	string (255)	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 **`stream`** structs	List of streams for which previews are required (maximum: 4) expressed as an array of stream structs.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
`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.

Stream struct

Required inputs

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

Parameter name	Type	Description
`streamIdentifier`	string	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.

Optional or conditional inputs

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

Parameter name	Type	Description
`position`	integer (>= 0 and <= {number of screens} - 1)	Required position of screen. For a single-screen endpoint, this should be 0. For a grouped or multi-screen endpoint, screen numbering starts at 0 going from left to right. For example, the right screen of a T3 has **`position`**= 2. Ignored if **`streamIdentifier`** is **`extendedVideo`**; must be supplied otherwise.
`maxWidth`	integer (>= 88)	Maximum width of generated preview. Default: 88.
`maxHeight`	integer (>= 72)	Maximum height of generated preview. Default: 72.

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.

Examples of circumstances that cause this method 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.

participantPreviewResponse

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
sourceIdentifier	string (255)	Source identifier provided in the original request (or Port A MAC address if not supplied in request).
streams	array of **preview** structs	Array of preview structs.

Preview struct

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

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**).

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.

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
message	string (500)	Message to display.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
position	integer (1–9)	Position on display: | 1 2 3 | | 4 5 6 | | 7 8 9 | Default: 5.
duration	integer (>= 0)	Duration, in seconds, for the message to display on the endpoint. Default: 30.

flex.participant.sendDTMF

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

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
dtmf	string (32)	The sequence of valid DTMF characters to send. See DTMF.

This method 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.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 method.

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
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.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.

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

flex.participant.clearImportant

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

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.

flex.participant.requestDiagnostics

Request call diagnostics for participants. If the participant has no active calls, this method fails with <Fault 56: 'no active participant call'>.

Input parameters

Required inputs

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

Parameter name	Type	Description
participantID	string (50)	Participant identifier assigned by the TelePresence Server. See Identifiers and client references.
receiverURI	string (255)	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).

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
sourceIdentifier	string (255) ASCII characters only	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). Each struct contains the following parameters and follows the standard XML-RPC specification.

The TelePresence Server can handle only a fixed number of diagnostics requests, so this method will fail with <Fault 203: 'too many asynchronous requests'> if the number of pending requests exceeds the limit.

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 audioRx stream struct.
audioTx	array	Array of **audioTx** stream structs. See audioTx stream struct.
auxiliaryAudioRx	array	Array of **auxiliaryAudioRx** stream structs. See auxiliary audioRx stream struct.

Parameter name	Type	Description
`auxiliaryAudioTx`	array	Array of `auxiliaryAudioTx` stream structs. See auxiliary audioTx stream struct.
`videoRx`	array	Array of `videoRx` stream structs. See videoRx stream struct.
`videoTx`	array	Array of `videoTx` stream structs. See videoTx stream struct.
`contentVideoRx`	array	Array of `contentRx` stream structs. See content videoRx stream struct.
`contentVideoTx`	array	Array of `contentTx` stream structs. See content videoTx stream struct.

`audioRx` stream struct

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

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, 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.

`audioTx` stream struct

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

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.

`auxiliaryAudioRx` stream struct

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

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, 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.

`auxiliaryAudioTx` stream struct

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

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.

`videoRx` stream struct

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

Parameter name	Type	Description
`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`.

Parameter name	Type	Description
`actualBitRate`	integer	Measured bit rate of this stream, in bits per second (bps).
`jitter`	integer	Current jitter in this stream, 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.
`fastUpdateRequestsSent`	integer	Number of fast update requests sent.
`muted`	boolean	Whether the stream is muted.

`videoTx` stream struct

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

Parameter name	Type	Description
`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.
`fastUpdateRequestsReceived`	integer	Number of fast update requests received.
`muted`	boolean	Whether the stream is muted.

`contentVideoRx` stream struct

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

Parameter name	Type	Description
`codec`	string	Codec in use.

Parameter name	Type	Description
height	integer	Height of the stream, in pixels.
width	integer	Width of the stream, in pixels.
encrypted	boolean	Whether the stream data is encrypted.
channelBitRate	integer	Bit rate of the channel in bits per second (bps).
expectedBitRate	integer	Expected bit rate of this stream, in bits per second (bps).
expectedBitRateReason	string	One of: **viewedSize**, **errorPackets**, or **notLimited**.
actualBitRate	integer	Measured bit rate of this stream, in bits per second (bps).
jitter	integer	Current jitter in this stream, 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.
fastUpdateRequestsSent	integer	Number of fast update requests sent.

contentVideoTx stream struct

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

Parameter name	Type	Description
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.
fastUpdateRequestsReceived	integer	Number of fast update requests received.

flex.call.status

Returns the status of the specified call.

Input parameters

Required inputs

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

Parameter name	Type	Description
callID	string (50)	Call identifier assigned by the TelePresence Server. See Identifiers and client references.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
callID	string (50)	Call identifier. See Identifiers and client references.
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.
conferenceState	string	State of call connection to the conference. See Call conference state.
callState	string	State of the call. See Call state.
incoming	boolean	Direction of the call: **true** indicates incoming; **false** indicates outgoing.
protocol	string	Call control protocol. See Protocol enumerated type.
address	string (80)	Address of the endpoint. For outgoing calls, this is the destination of the call.

Conditionally returned

The following table lists the parameters that are conditionally returned by this method.

Parameter name	Type	Description
participantID	string (50)	Participant identifier. See Identifiers and client references.
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.

Parameter name	Type	Description
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.

system.info

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

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
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.
numControlledServers	integer	Number of TelePresence Servers controlled by this unit (including itself).
operationMode	string	One of **standalone** (locally managed), **flexible** (remotely managed), or **slave** (slave blade in a cluster).
licenseMode	string	If **operationMode** = **standalone**: one of **HD** or **fullHD**. If **operationMode** = **flexible**: always **flexible**. If **operationMode** = **slave**: not present.
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.

Parameter name	Type	Description
`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.
`maxConferenceSizeContent`	integer	In flexible (remotely managed) mode, this value is always 0 and should be ignored.

device.query

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

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`currentTime`	dateTime.iso8601	Current time (UTC) on the system.
`restartTime`	dateTime.iso8601	Date and time at which the system was last restarted.
`uptime`	integer	Number of seconds since the last restart.
`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 contains a string named **`feature`** containing a short description of that feature, for example, **`Encryption`**.
`shutdownStatus`	string	Displays one of the following: **`notShutdown`**, **`shutdownInProgress`**, **`shutdown`**, or **`error`**.

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.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
portA	port struct	A structure that contains configuration and status information for Ethernet port A on the device.
dns	Array of dns structs	An array whose members represent the DNS parameters of the device.

Port struct

Returned data

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

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. See Note.
packetsReceived	integer	Number of packets received on this Ethernet port. See Note.
multicastPacketsSent	integer	Number of multicast packets sent from this Ethernet port. See Note.
multicastPacketsReceived	integer	Number of multicast packets received on this Ethernet port. See Note.
bytesSent	integer	Number of bytes sent by the device. See Note.
bytesReceived	integer	Number of bytes received by the device. See Note.
queueDrops	integer	Number of packets dropped from the queue on this network port. See Note.
collisions	integer	Count of the network collisions recorded by the device. See Note.
transmitErrors	integer	Count of transmission errors on this Ethernet port. See Note.
receiveErrors	integer	Count of receive errors on this port. See Note.

Parameter name	Type	Description
`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.

Note

All interface statistic values are 32-bit signed integers, and therefore may wrap.

Conditionally returned

The following parameters are returned only if the interface is enabled and configured.

Parameter name	Type	Description
`dhcpv4`	boolean	Whether the ipv4 address is allocated by DHCP.
`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.
`ipv6Address`	string	IPv6 address in CIDRformat.
`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).
`ipv6PrefixLength`	integer	Length of the IPv6 address prefix.
`defaultIpv6Gateway`	string	Address of the IPv6 default gateway in CIDR format.
`linkLocalIpv6Address`	string	Link local IPv6 address in CIDR format.
`linkLocalIpv6PrefixLength`	integer	Length of the link local IPv6 address prefix.

dns struct

Returned data

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

Parameter name	Type	Description
`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.restartlog.query

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

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`log`	array of `log` structs	Each member of the array contains log information (called system log in the user interface).

Log struct

Returned data

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

Parameter name	Type	Description
`time`	dateTime.iso8601	Date and time of the last reboot.
`reason`	string	Reason for the device restart. See Log reason enumerated types.

Log reason enumerated types

The following table lists the log reason enumerated types.

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

device.restart

Restarts the device, or shuts it down without a restart.

Input parameters

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
`shutdownOnly`	boolean	Whether the device shuts down without restarting upon receipt of the request. Default: **`false`**.

device.health.query

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

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`cpuLoad`	integer	CPU load expressed as a percentage of the maximum.
`fanStatus`	string	One of **`ok`** or **`outOfSpec`**. This parameter is returned only on 7010 devices, not 8710s.
`fanStatusWorst`	string	Worst fan status recorded on this device since it rebooted. One of **`ok`** or **`outOfSpec`**. This parameter is returned only on 7010 devices, not 8710s.
`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`**.

Parameter name	Type	Description
operationalStatus	string	One of **active** (the device is active), **shuttingDown** (the device is shutting down), or **shutDown** (the device has shut down).

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.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

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.
numEvents	integer	Maximum number of events to be returned per enumeration. If omitted (or not between 1–20 inclusive), up to 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**.

Returned data

The following table lists the parameters that are returned by this method.

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.

Parameter name	Type	Description
`eventsRemaining`	boolean	Whether there is data remaining after this enumeration.
`currentTime`	dateTime.iso8601	The system's current time (UTC).
`events`	array (see Events array)	List of the new events; these are structures with some common fields (time, type, index) and other fields specific to the event type.

Events array

The following parameters are common to all CDR log events. The array also contains information members specific to each event. The CDR log reference guide[3] contains details of the TelePresence Server event types.

Note: The CDR log 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.

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

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.

cdrlog.query

Returns high level status information about the CDR log. This method takes no input parameters.

Returned data

The following table lists the parameters that are returned by this method.

Parameter name	Type	Description
`firstIndex`	integer	Index of the oldest stored event.
`numEvents`	integer	Total number of events stored.

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.

Returned data

The following table lists the parameters that are returned by this method.

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.

feedback receiver details

The following table lists the parameters that are returned by this array.

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.configure

Input parameters

Required inputs

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

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

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
`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.

Parameter name	Type	Description
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. If this array is absent, the receiver subscribes to all notifications by default. Default: all events.

Returned data

The following table lists the parameters that are returned by this method.

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

feedbackReceiver.reconfigure

Changes the configuration of an existing feedback receiver. The current configuration is only changed for parameters that are included in the request.

Required inputs

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

Parameter name	Type	Description
receiverIndex	integer (1–20)	Index of the feedback receiver to be reconfigured. The call returns a fault if there is no feedback receiver at the specified **receiverIndex**.

Optional or conditional inputs

The following table lists the optional or conditional input parameters that are accepted by this method.

Parameter name	Type	Description
receiverURI	string (255)	Fully-qualified **http** or **https** URI (for example, **http://tms1:8080/RPC2**) to which feedback events are sent. If no port number is specified, the device uses the protocol defaults (80 and 443 respectively).
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	Array of strings identifying the events to which the receiver subscribes. See Feedback events. If this array is absent, the event notifications set in the original configuration request remain unchanged. Default: all events.

feedbackReceiver.remove

Removes the specified feedback receiver.

Input parameters

Required inputs

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

Parameter name	Type	Description
`receiverIndex`	integer (1–20)	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.

Required inputs

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

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

Returned data

The following table lists the parameters that are returned by this method.

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.

Related information

system.xml

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.

System XML contents

Node name	Node contents
`gatekeeperUsage`	**`Yes`**: gatekeeper usage is enabled. **`No`**: gatekeeper usage is disabled.
`gatekeeperAddress`	Gatekeeper 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.
`sipRegistrarAddress`	SIP registrar IP address.
`sipRegistrarDomain`	SIP registrar domain name.
`sipTrunkUsage`	**`Yes`**: trunk usage is enabled. **`No`**: trunk usage is disabled.
`sipTrunkAddress`	SIP trunk IP address.
`sipTrunkDomain`	SIP trunk domain name.
`isMaster`	**`Yes`**: this system is a master. **`No`**: this system is a slave.
`clusterType`	Role of this system in a cluster: one of **`unclustered`**, **`master`**, or **`slave`**.
`totalVideoPorts`	Total number of video ports supported by the device. In flexible (remotely managed) mode, this value is always 0 and should be ignored.
`totalContentPorts`	Total number of content ports supported by the device. In flexible (remotely managed) mode, this value is always 0 and should be ignored.
`totalAudioOnlyPorts`	Total number of audio only ports supported by the device. In flexible (remotely managed) mode, this value is always 0 and should be ignored.
`uptimeSeconds`	Length of time in seconds that the device has been up and running.

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>
```

References

1. XML-RPC specification (Dave Winer, June 1999); http://www.xmlrpc.com/spec, accessed 24/01/2011.
2. HTTP/1.1 specification (RFC 2616, Fielding et al., June 1999); http://www.ietf.org/rfc/rfc2616.txt, accessed 24/01/2011.
3. Cisco TelePresence CDR logs reference guide (Cisco Systems Inc., June 2011); http://www.cisco.com/en/US/docs/telepresence/infrastructure/mcu/admin_guide/Cisco_TelePresence_infrastructure_products_CDR_log_reference.pdf, accessed 07/07/2011.

Part 2: Standalone operation mode

Introduction

This document accompanies the latest version of the remote management API for the Cisco TelePresence Server software (respectively referred to as API and TelePresence Server in this document). The following Cisco TelePresence products support this API when they are running TelePresence Server version 3.0 and later:

- Cisco TelePresence Server MSE 8710
- Cisco TelePresence Server 7010

XML-RPC implementation

API calls and responses are implemented using the XML-RPC protocol. This simple protocol does remote procedure calling using HTTP (or HTTPS) as the transport and XML as the encoding, however, it still allows for complex data structures. XML-RPC is stateless and is not platform-dependent; it was chosen in favor of SOAP (Simple Object Access Protocol) because of its simplicity.

Your application must either regularly poll the device or continually listen to the device - if it is configured to publish feedback events - if you want it to monitor the device's activity.

The API implements all parameters and returned data as **`<struct>`** elements, each of which is explicitly named. For example, **`device.query`** returns (amongst other data) the current time as:

```
<member>
  <name>currentTime</name>
  <value><dateTime.iso8601>20110121T13:31:26<dateTime.iso8601></value>
</member>
```

rather than simply

```
<dateTime.iso8601>20110121T13:31:26<dateTime.iso8601>
```

Note: Unless otherwise stated, assume strings have a maximum length of 31 characters.

Refer to the XML-RPC specification[1] for more information.

Transport protocol

The device implements HTTP/1.1 as defined by RFC 2616[2]. 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.

Considering API overhead when writing applications

Every API command that your application sends incurs a processing overhead within the device’s own application. The exact amount of overhead varies widely with the command type and the parameters sent. It is important to bear this in mind when designing your application’s architecture and software. If the device receives a high number of API commands every second, its overall performance could be seriously impaired – in the same way that it would be if several users accessed it simultaneously via the web interface.

There is a limit on the number of simultaneous requests that the TelePresence Server can process. If your client application receives a related HTTP error such as `503 Service Unavailable` then it should retry the request.

For this reason, the best architecture is a single server running the API application and sending 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. The server would then 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 commands. This provides much more control than having the clients send API commands directly and will prevent the device's performance being impaired by unmanageable numbers of API requests.

Furthermore, the API is designed to have as little impact as possible on the network when responding to requests. The device's responses do not routinely include data that is not relevant, or empty data structures where the data is not available. Your application should take responsibility for checking whether the response includes what you expected, and you should design it to gracefully handle any situations where the device does not respond with the expected data.

Standalone operation mode API Change Summary

The latest Cisco TelePresence Server API is version 3.0. The table below is a summary of the changes to the standalone operation mode API from version 2.3 to 3.0.

XML-RPC Request	Parameter	Change
system.info	`operationMode` `licenseMode`	Modified

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.

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.

- Use `feedbackReceiver.configure [p.148]` to register a receiver to listen for one or more feedback events.
- Use `feedbackReceiver.query [p.147]` to return a list of receivers that are configured on the device.
- Use `feedbackReceiver.reconfigure [p.149]` to change the configuration of an existing feedback receiver.
- Use `feedbackReceiver.remove [p.151]` to remove an existing feedback receiver.
- Use `feedbackReceiver.status [p.151]` to view a list of the events that a feedback receiver is subscribed to.

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 otherwise 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.

Event	Description
restart	The source publishes this event when it starts up
configureAck	The source publishes this event to acknowledge that an application has successfully configured a feedback receiver
cdrAdded	One or more new Call Detail Records have been logged
conferenceStarted	One or more conferences have been created
conferenceFinished	One or more conferences have been deleted

Event	Description
conferenceActive	One or more conferences have become active (first participant joined)
conferenceInactive	One or more conferences have become inactive (last participant left)
participantJoined	One or more participants have joined a conference
participantLeft	One or more participants have left a conference
participantConnected	One or more participants have connected to the TelePresence Server
participantDisconnected	One or more participants disconnected from the TelePresence Server
receiverModified	The feedback receiver receiving this event has been modified. Cannot be unsubscribed from or disabled.
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. Cannot be unsubscribed from or disabled.
deviceStatusChanged	Generated when the TelePresence Server is shut down or a feature key is added or removed. Should result in a device.query request being made.

API overview

Encoding

Your application can encode messages as ASCII text or as UTF-8 Unicode. If you do not specify the encoding, the API assumes ASCII encoding. You can specify the encoding in a number of ways:

Specify encoding with HTTP headers

There are two ways of specifying UTF-8 in the HTTP headers:

- Use the **`Accept-Charset: utf-8`** header
- Modify the **`Content-Type`** header to read **`Content-Type: text/xml; charset=utf-8`**

Specify encoding with XML header

The **`<?xml>`** tag is required at the top of each XML file. The API will accept an encoding attribute for this tag; that is, **`<?xml version="1.0" encoding="UTF-8"?>`**.

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 authentication parameters:

`authenticationUser`

Type: **string**

Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.

`authenticationPassword`

Type: **string**

The password that corresponds with the given **`authenticationUser`**. The API ignores this parameter if the user has no password. This behavior differs from the web interface, where a blank password must be blank.

Message flow

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

The example command below is: create conference: 'API Conference' with numeric ID: '971771' and PIN: '123'

Example command

```
<?xml version='1.0'?>
```

```
<methodCall>
  <methodName>conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>authenticationPassword</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>API Conference</string>
            </value>
          </member>
          <member>
            <name>numericID</name>
            <value>
              <string>971771</string>
            </value>
          </member>
          <member>
            <name>PIN</name>
            <value>
              <string>123</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:

- With an XML **`methodResponse`** message that may or may not contain data, depending on the command.
- With an XML **`methodResponse`** that includes only a fault code message.

Example success

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
```

```
      <value>
        <struct>
          <member>
            <name>conferenceID</name>
            <value>
              <int>10000</int>
            </value>
          </member>
          <member>
            <name>conferenceGUID</name>
            <value>
              <string>62f46be0-c6a3-11e1-9800-000d7c10cc70</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Example fault code

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value>
            <int>13</int>
          </value>
        </member>
        <member>
          <name>faultString</name>
          <value>
            <string>invalid PIN</string>
          </value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

API reference

This is a list of the API calls supported by the TelePresence Server. For each API call, the following information is provided where applicable:

- Description of the call's function and status
- Accepted parameters
- Returned parameters, structure formats and data types
- Deprecated parameters

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

- cdrlog.enumerate
- cdrlog.query
- conference.create
- conference.delete
- conference.enumerate
- conference.invite
- conference.senddtmf
- conference.sendmessage
- conference.sendwarning
- conference.set
- conference.status
- conference.uninvite
- device.health.query
- device.network.query
- device.query
- device.restartlog.query
- device.restart
- feedbackReceiver.configure
- feedbackReceiver.query
- feedbackReceiver.reconfigure
- feedbackReceiver.remove
- feedbackReceiver.status
- participant.diagnostics
- participant.enumerate
- participant.set
- participant.tidylayout
- system.info

Deprecations

Deprecated parameters

The following parameters were deprecated in version 2.2 of the API. Update your applications to use the replacement parameters instead; these parameters may not be supported in future releases.

In calls that can still accept the deprecated parameters, take care to send only the deprecated parameter or the replacement parameter; not both.

Deprecated parameter	Replaced by	The affected calls
permanent	persistent	conference.create
conferenceID	conferenceGUID	conference.invite conference.create conference.delete conference.enumerate conference.senddtmf conference.sendmessage conference.sendwarning conference.set conference.status conference.uninvite participant.enumerate
participantList (string)	participants (array)	conference.invite
participantID	participantGUID	conference.invite conference.senddtmf conference.sendmessage conference.status participant.enumerate participant.set participant.tidylayout
omitID	omitGUID	conference.senddtmf
endpointType	endpointCategory	conference.status
participantListID	participantListGUID	conference.uninvite
roundTableEnable	oneTableMode	conference.set conference.status

cdrlog.enumerate

Status: **active**

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.

The 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.

Accepts:

Optional:

`filter`

Type: **array**

An array of strings, each of which is the name of an event type by which to filter the response. If the array is omitted, all event types are returned.

For the TelePresence Server, the call can request any / all of the following event types:

- conferenceStarted
- conferenceActive
- participantConnected
- participantJoined
- participantMediaSummary
- participantLeft
- participantDisconnected
- conferenceInactive
- conferenceFinished

`index`

Type: **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, then the device will enumerate events from the beginning of the CDR log.

`numEvents`

Type: **integer**

Specifies 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.

Returns:

The response provides reference information such as time and log position, and an array of events that meet the parameters provided in the call.

`startIndex`

Type: **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`

Type: **integer**

Revision number of the data being provided, reusable in a subsequent call to the API.

`eventsRemaining`

Type: **boolean**

Whether there is data remaining after this. Provided to avoid putting all data in a single call.

`currentTime`

Type: **dateTime.iso8601**

The system's current time (UTC).

`events`

Type: **array**

List of the new events; these are structures with some common fields (time, type, index) and other fields specific to the event type.

`events` array

The following parameters are common to all CDR log events. The array also contains information members specific to each event. The CDR log reference guide[3] contains details of the TelePresence Server event types.

Note: The CDR log 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.

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

`time`

Type: **dateTime.iso8601**

The date and time when the event was logged.

Value	Description
20110119T13:52:42	yyyymmddThh:mm:ss

`type`

Type: **string**

The name of the event type.

`index`

Type: **integer**

A number that identifies the position of the item in context with similar items.

cdrlog.query

Status: **active**

This call queries for statistics about the CDR log.

This call takes no parameters.

Returns:

`firstIndex`

Type: **integer**

The index of the oldest stored event.

`numEvents`

Type: **integer**

The total number of events stored.

conference.create

Status: **active**

Creates a conference with the specified name and other supplied parameters, and returns the unique identifier of the new conference.

Accepts:

Required:

`conferenceName`

Type: **string** (up to 80 characters)

The name that refers to the conference that is the subject of your call or the response from the TelePresence Server.

Optional:

Use **`persistent`** instead of **`permanent`** to define conference persistence, even though the TelePresence Server will accept either.

`persistent`

Type: **boolean**

Defines whether the conference persists after all participants leave. Persistent conferences are stored in the configuration file and thus will persist through a device restart.

Value	Description
true	The conference persists, irrespective of participants leaving, until it is explicitly deleted.
false	The conference is deleted 30 seconds after all participants have left, or when **`duration`** expires (if it is set).

`permanent`

Type: **boolean**

Deprecated. Use **`persistent`** instead.

Defines whether the conference persists after all participants leave. Without this option any conferences will be automatically deleted after 30 seconds, or when **`duration`** expires (if it is set).

`locked`

Type: **boolean**

Defines whether the conference is locked.

Endpoints can not join a locked conference but the conference can invite them in.

Value	Description
true	The conference is locked.
false	The conference is not locked.

`lockDuration`

Type: **integer**

The period of time (in seconds) from now until the conference lock expires. Requires that **`locked`** is **`true`** and ignored otherwise.

`numericID`

Type: **string** (up to 80 characters)

Used for registration with H.323 gatekeeper / SIP registrar, and to dial in to the conference.

`registerWithGatekeeper`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the H.323 gatekeeper.

`registerWithSIPRegistrar`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the SIP registrar.

`tsURI`

Type: **string** (up to 80 characters)

The address that Cisco TelePresence System T3 systems use to make API calls to the TelePresence Server.

This string must take the form **`[<protocol>://]<address>[:<port>]`**, for example, **`http://mytps:80`**. If supplied, this URI will be passed to all T3 systems in the conference via TString. If not explicitly supplied, the TelePresence Server will create a **`tsURI`** based on its IP address.

`http` and **`https`** protocols are supported. The TelePresence Server does not assume protocol or port information if the application does not supply them in this string.

`h239ContributionEnabled`

Type: **boolean**

Defines whether the conference allows content contribution. This parameter controls whether content may be contributed via any of the supported content protocols; it is not limited to H.239.

`useLobbyScreen`

Type: **boolean**

Defines whether the conference shows the lobby screen.

`lobbyMessage`

Type: **string** (up to 500 characters)

The lobby screen message.

`useWarning`

Type: **boolean**

Defines whether the conference sends 'This conference is about to end' warning.

`audioPortLimit`

Type: **integer**

The limit on the number of audio ports this conference may allow.

`videoPortLimit`

Type: **integer**

The limit on the number of video ports this conference may allow.

`duration`

Type: **integer**

Period of time (in seconds) until the conference ends and is deleted.

This parameter is not allowed if **`persistent`** is **`true`**.

`pin`

Type: **string** (up to 40 characters)

The PIN for this conference. If associated with a conference, it is a string of numeric digits that must be entered to gain access to that conference.

Note: A PIN is only valid for incoming calls—no outgoing calls will ever need to enter it. As a result of this, a conference PIN can only be set when the conference has a numeric ID. Trying to set a PIN without a numeric ID will return a fault, and clearing a conference's numeric ID will also clear that conference's PIN.

Returns:

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use **`conferenceGUID`** instead.

Unique conference identifier.

conference.delete

Status: **active**

Deletes the specified conference.

Accepts:

Required:

To identify the conference, use **`conferenceGUID`** instead of **`conferenceID`**, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use **`conferenceGUID`** instead.

Unique conference identifier.

conference.enumerate

Status: **active**

Requests information about all the conferences on the TelePresence Server. The full enumeration response may require multiple calls.

Accepts:

Optional:

`enumerateID`

Type: **integer**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of results. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

`activeFilter`

Type: **boolean**

Value	Description
true	Request only active conferences
false	Request all conferences. This is the default value; it is assumed if you omit the parameter.

Returns:

If there are no conferences to enumerate, then the **`conference.enumerate`** call does not return the **`conferences`** array.

Conditional:

`enumerateID`

Type: **integer**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of results. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

`conferences`

Type: **array**

An array of structs, each of which contains all the returned information about a single conference.

`conferences` array members

The following information is returned about the enumerated conferences:

`conferenceName`

Type: **string** (up to 80 characters)

The name that refers to the conference that is the subject of your call or the response from the TelePresence Server.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use `conferenceGUID` instead.

Unique conference identifier.

The TelePresence Server returns this parameter, even though it is deprecated, to ensure application compatibility in the short term. We recommend that you use the replacement parameter instead.

`active`

Type: **boolean**

Value	Description
true	The conference is currently active
false	The conference is currently inactive (e.g. a persistent conference without any active participants, or a non-persistent conference that has not yet started or is empty but not yet deleted)

`persistent`

Type: **boolean**

Defines whether the conference persists after all participants leave. Persistent conferences are stored in the configuration file and thus will persist through a device restart.

Value	Description
true	The conference persists, irrespective of participants leaving, until it is explicitly deleted.
false	The conference is deleted 30 seconds after all participants have left, or when `duration` expires (if it is set).

`locked`

Type: **boolean**

Defines whether the conference is locked.

Endpoints can not join a locked conference but the conference can invite them in.

Value	Description
true	The conference is locked.
false	The conference is not locked.

`numericID`

Type: **string** (up to 80 characters)

Used for registration with H.323 gatekeeper / SIP registrar, and to dial in to the conference.

This is an empty string if the parameter is not set.

`registerWithGatekeeper`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the H.323 gatekeeper.

`registerWithSIPRegistrar`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the SIP registrar.

`h239ContributionEnabled`

Type: **boolean**

Defines whether the conference allows content contribution. This parameter controls whether content may be contributed via any of the supported content protocols; it is not limited to H.239.

`pin`

Type: **string** (up to 40 characters)

The PIN for this conference. If associated with a conference, it is a string of numeric digits that must be entered to gain access to that conference.

Note: A PIN is only valid for incoming calls—no outgoing calls will ever need to enter it. As a result of this, a conference PIN can only be set when the conference has a numeric ID. Trying to set a PIN without a numeric ID will return a fault, and clearing a conference’s numeric ID will also clear that conference’s PIN.

conference.invite

Status: **active**

Invites the specified participants to the specified conference.

Avoid using the **`conferenceID`** and **`participantList`** parameters and use the replacement **`conferenceGUID`** and **`participants`** parameters instead.

Accepts:

Required:

To identify the conference, use **`conferenceGUID`** instead of **`conferenceID`**, not both.

To identify the participants, use the **`participants`** array instead of **`participantList`**, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use **`conferenceGUID`** instead.

Unique conference identifier.

`participantList`

Type: **string**

Deprecated. Use **`participants`** array instead.

A comma separated list of participant addresses, with optional extra information.

Value	Description
Example	**`10.2.171.232, 10.47.2.246, h323:numericID@domain.com`**
Example with type	**`10.2.171.232, t3:h323:numericID@domain.com`** (specify the endpoint type,followed by a colon, before the protocol)
Example with master	**`10.2.171.232, t3:master:h323:numericID@domain.com`** (specify **`master:`** in the prefix; immediately after the endpoint type, if present, and before the protocol)

`participants`

Type: **array**

An array of structures that represent participants.

`participants` array

You must include an array of participants in your **`conference.invite`** call. Each participant must have an **`address`** parameter. All participant parameters except **`address`** are optional and the TelePresence Server will use the default value if your call omits them.

`address`

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or E.164 number.

You must prefix the address with either **`h323:`** or **`sip:`**. If you do not provide a prefix, the TelePresence Server attempts to call the address directly, using H.323 (not via the gatekeeper). The maximum length of the address is 80 characters (note that prefixes such as **`h323:`** are included in this limit).

You may provide a comma separated list of up to four addresses if you are inviting a grouped endpoint (requires a third-party interop feature key installed on the TelePresence Server). In this case you should provide a protocol prefix for each address, for example **`h323:leftmost_endpoint@domain.com,h323:rightmost_endpoint@domain.com`**, but must not supply a type prefix.

The maximum length of each address is 80 characters (note that prefixes such as **`h323:`** are included in this limit).

The total length of the value supplied (up to four addresses and separating commas) cannot exceed 323 characters.

type

Type: **string**

Specifies the type of endpoint.

Value	Description
t3	Cisco TelePresence System T3
cts	Any Cisco TelePresence System 'telepresence' endpoint (1 or 3 screen, e.g. 500, 1300, 3000)
cts1	Cisco TelePresence System single screen 'telepresence' endpoints (e.g. 500 and 1300 series)
cts3	Cisco TelePresence System three screen 'telepresence' endpoints (e.g. 3000 series)

master

Type: **boolean**

Value	Description
true	This endpoint is conference master
false	(default if omitted) This endpoint is not the conference master

oneTableIndex

Type: **integer**

The endpoint's position if it is in a OneTable conference. Applies only if **`type`** is **`t3`**.

Value	Description
1, 2, 3, or 4	Position index of the endpoint when it is in OneTable mode. The positions increment around the one virtual table in a clockwise manner, when the table is viewed from above. For example, the participant whose index is 2 will appear to be sitting to the left of the participant whose index is 1.

maxBitRate

Type: **integer**

The maximum bitrate, in kbps, in both directions between the TelePresence Server and this participant. The TelePresence Server uses its default setting if your call omits this parameter.

`recordingDevice`

Type: **boolean**

Value	Description
true	The endpoint is treated as a recording device; it does not feature in the layout and other participants are made aware of its presence by a red dot as appropriate.
false	(default if omitted) The endpoint is a normal endpoint.

`dtmf`

Type: **string** (up to 50 characters)

DTMF character string to send to this endpoint after connection.

`audioContentIndex`

Type: **integer**

Defines which endpoint in a group should receive the content and audio. This is a zero-based index that corresponds to the entries provided in the comma separated list of endpoint addresses in **`address`**.

Value	Description
0	(default) The first address in the address string
n-1	(maximum) The last address in a comma separated string of n addresses

`contentIndex`

Type: **integer**

Defines which endpoint in a group should receive the content (if different to **`audioContentIndex`**). It is ignored unless **`audioContentIndex`** is supplied in the request.This is a zero-based index that corresponds to the entries provided in the comma separated list of endpoint addresses in **`address`**. Defaults to **`audioContentIndex`**.

Value	Description
0	(default) The first address in the address string
n-1	(maximum) The last address in a comma separated string of n addresses

Use this parameter if the endpoint sending/receiving content is different to that sending/receiving audio (specified in **`audioContentIndex`**).

`camerasCrossed`

Type: **boolean**

Value	Description
true	The cameras of a grouped endpoint are crossed; this is ignored unless this participant is a grouped endpoint, i.e. has multiple **`address`** parameters
false	(default if omitted) The cameras are not crossed

`txAspectRatio`

Type: **string**

Overrides the aspect ratio of the layout transmitted to this participant.

Value	Description
only16to9	Force the TelePresence Server to send a widescreen layout (16:9) to the endpoint, overriding any box-wide or per-endpoint settings
only4to3	Force the TelePresence Server to send a 4:3 layout to the endpoint, overriding any box-wide or per-endpoint settings

`autoReconnect`

Type: **boolean**

Defines whether the TelePresence Server attempts to re-establish the call to this endpoint (or a member of a group if the endpoint is grouped), if it fails or disconnects due to an error.

Value	Description
True	Five retries will be attempted at intervals of 5 seconds (first retry after failure/disconnection), 15 seconds, 30 seconds, 60 seconds, 120 seconds.
False	The TelePresence Server will not attempt to reconnect the call. This is the default.

Returns:

`participantList`

Type: **array**

Array of participants. Each member of the array is a struct that represents a participant on the TelePresence Server.

`participantList` array members

The returned **`participantList`** is an array of successfully invited participants. Note that the member structs of this array are different to those returned in **`participantList`** by the **`conference.status`** call.

Each struct contains the following parameters:

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`participantID`

Type: **integer**

Deprecated. Use **`participantGUID`** instead.

The unique ID of this participant, assigned by the TelePresence Server.

The TelePresence Server returns this parameter, even though it is deprecated, to ensure application compatibility in the short term. We recommend that you use the replacement parameter instead.

`address`

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or E.164 number.

These addresses are as you supplied them in the participants array, to make them easier to compare.

conference.senddtmf

Status: **active**

Sends a DTMF string to some or all participants in the specified conference. You must specify the conference and the DTMF string (up to 50 characters).

If you don't specify a participant, the string goes to all participants; otherwise, you may specify either a participant who will receive the string or one who will not receive the string.

Accepts:

Required:

To identify the conference, use **`conferenceGUID`** instead of **`conferenceID`**, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use **`conferenceGUID`** instead.

Unique conference identifier.

`dtmf`

Type: **string** (up to 50 characters)

DTMF character string to send to this endpoint after connection.

Optional:

To identify the participant to receive DTMF, use **`participantGUID`** instead of **`participantID`**, not both. Alternatively, to identify the participant who won't receive DTMF, use **`omitGUID`** instead of **`omitID`**, not both.

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

If you supply this parameter, the DTMF string will be sent to this participant only.

`participantID`

Type: **integer**

Deprecated. Use **`participantGUID`** instead.

The unique ID of this participant, assigned by the TelePresence Server.

`omitGUID`

Type: **string**

A **`participantGUID`**. Prevents this participant from receiving the DTMF string specified in **`dtmf`**.

If you supply this parameter, the DTMF string will be sent to all participants except this one. If **`participantGUID`** is present, **`omitGUID`** is ignored.

`omitID`

Type: **integer**

Deprecated. Use **`omitGUID`** instead.

A **`participantID`**. Prevents this participant from receiving the DTMF string specified in **`dtmf`**.

conference.sendmessage

Status: **active**

Sends a message to all participants in the specified conference. You must specify the conference and the message.

If you choose to specify a participant, the message will only go to that participant.

Accepts:

Required:

To identify the conference, use **`conferenceGUID`** instead of **`conferenceID`**, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use `conferenceGUID` instead.

Unique conference identifier.

`message`

Type: **string** (up to 500 characters)

Message to send to conference.

Optional:

To identify a participant, use the `participantGUID` instead of `participantID`, not both.

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`participantID`

Type: **integer**

Deprecated. Use `participantGUID` instead.

The unique ID of this participant, assigned by the TelePresence Server.

`position`

Type: **integer**

Defines where the message displays on the layout.

Value	Description
`1`,`2`, or `3`	The message displays near the top of the layout; aligned to the left, center, or right respectively.
`4`, `5` (default), or `6`	The message displays in the middle of the layout; aligned to the left, center, or right respectively.
`7`, `8`, or `9`	The message displays near the bottom of the layout; aligned to the left, center, or right respectively.

`duration`

Type: **integer**

Period of time (in seconds) for which the message is displayed to participants. Default is `30`.

conference.sendwarning

Status: **active**

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

Accepts:

Required:

To identify the conference, use `conferenceGUID` instead of `conferenceID`, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use `conferenceGUID` instead.

Unique conference identifier.

Optional:

`secondsRemaining`

Type: **integer**

The number of seconds from now in which the conference will end.

This value is used when informing CTS endpoints (using CCCP) that the conference is ending.

conference.set

Status: **active**

Edit the configuration of the specified conference.

Accepts:

Required:

To identify the conference, use `conferenceGUID` instead of `conferenceID`, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use `conferenceGUID` instead.

Unique conference identifier.

Optional:

To set up one table mode, use **`oneTableMode`** instead of **`roundTableEnable`**, not both.

`numericID`

Type: **string** (up to 80 characters)

Used for registration with H.323 gatekeeper / SIP registrar, and to dial in to the conference.

`registerWithGatekeeper`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the H.323 gatekeeper.

`registerWithSIPRegistrar`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the SIP registrar.

`roundTableEnable`

Type: **boolean**

Deprecated. Use **`OneTableMode`** instead.

Defines whether the conference is in round table mode.

If you supply both **`roundTableEnable`** and **`OneTableMode`**, then the TelePresence Server will use **`OneTableMode`** without returning an error.

`oneTableMode`

Type: **integer**

Value	Description
0	OneTableMode off
1	4 person OneTableMode

`h239ContributionEnabled`

Type: **boolean**

Defines whether the conference allows content contribution. This parameter controls whether content may be contributed via any of the supported content protocols; it is not limited to H.239.

`locked`

Type: **boolean**

Defines whether the conference is locked.

Endpoints can not join a locked conference but the conference can invite them in.

Value	Description
true	The conference is locked.
false	The conference is not locked.

`lockDuration`

Type: **integer**

The period of time (in seconds) from now until the conference lock expires. Requires that **`locked`** is **`true`** and ignored otherwise.

`duration`

Type: **integer**

Period of time (in seconds) until the conference ends and is deleted.

This parameter is not allowed if **`persistent`** is **`true`**.

You can pass a negative value to clear a previously set **`duration`**.

`audioPortLimitSet`

Type: **boolean**

Defines whether the **`audioPortLimit`** is applied.

Value	Description
true	Limits the number of audio ports to the value in **`audioPortLimit`**
false	**`audioPortLimit`** is ignored if it is present

You **must** provide an **`audioPortLimit`** if you set **`audioPortLimitSet`** to **`true`**. If you set it **`false`**, the call clears the existing **`audioPortLimit`**.

`audioPortLimit`

Type: **integer**

The limit on the number of audio ports this conference may allow.

`videoPortLimitSet`

Type: **boolean**

Defines whether the **`videoPortLimit`** is applied.

Value	Description
true	Limits the number of video ports to the value in **`videoPortLimit`**
false	**`videoPortLimit`** is ignored if it is present

You **must** provide a **`videoPortLimit`** if you set **`videoPortLimitSet`** to **`true`**. If you set it **`false`**, the call clears the existing **`videoPortLimit`**.

`videoPortLimit`

Type: **integer**

The limit on the number of video ports this conference may allow.

`useLobbyScreen`

Type: **boolean**

Defines whether the conference shows the lobby screen.

`lobbyMessage`

Type: **string** (up to 500 characters)

The lobby screen message.

`useWarning`

Type: **boolean**

Defines whether the conference sends 'This conference is about to end' warning.

`pin`

Type: **string** (up to 40 characters)

The PIN for this conference. If associated with a conference, it is a string of numeric digits that must be entered to gain access to that conference.

Note: A PIN is only valid for incoming calls—no outgoing calls will ever need to enter it. As a result of this, a conference PIN can only be set when the conference has a numeric ID. Trying to set a PIN without a numeric ID will return a fault, and clearing a conference's numeric ID will also clear that conference's PIN.

conference.status

Status: **active**

Reports the current status of the specified conference and its participants.

Accepts:

Required:

To identify the conference, use **`conferenceGUID`** instead of **`conferenceID`**, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use **`conferenceGUID`** instead.

Unique conference identifier.

Optional:

`enumerateID`

Type: **integer**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of results. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

Returns:

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use **`conferenceGUID`** instead.

Unique conference identifier.

The TelePresence Server returns this parameter, even though it is deprecated, to ensure application compatibility in the short term. We recommend that you use the replacement parameter instead.

`active`

Type: **boolean**

Value	Description
true	The conference is currently active
false	The conference is currently inactive (e.g. a persistent conference without any active participants, or a non-persistent conference that has not yet started or is empty but not yet deleted)

`persistent`

Type: **boolean**

Defines whether the conference persists after all participants leave. Persistent conferences are stored in the configuration file and thus will persist through a device restart.

Value	Description
true	The conference persists, irrespective of participants leaving, until it is explicitly deleted.
false	The conference is deleted 30 seconds after all participants have left, or when **`duration`** expires (if it is set).

`duration`

Type: **integer**

Period of time (in seconds) until the conference ends and is deleted.

This parameter is not allowed if **`persistent`** is **`true`**.

`locked`

Type: **boolean**

Defines whether the conference is locked.

Endpoints can not join a locked conference but the conference can invite them in.

Value	Description
true	The conference is locked.
false	The conference is not locked.

`lockDuration`

Type: **integer**

The period of time (in seconds) from now until the conference lock expires. Requires that **`locked`** is **`true`** and ignored otherwise.

`roundTableEnable`

Type: **boolean**

Deprecated. Use **`OneTableMode`** instead.

Defines whether the conference is in round table mode.

The TelePresence Server returns this parameter, even though it is deprecated, to ensure application compatibility in the short term. We recommend that you use the replacement parameter instead.

`oneTableMode`

Type: **integer**

Value	Description
0	OneTableMode off
1	4 person OneTableMode

`h239ContributionID`

Type: **integer**

The **`participantID`** of the endpoint that is contributing H.239 content. Zero if there is no H.239 contribution.

`portsVideoFree`

Type: **integer**

Count of the currently unused video ports.

Zero if the conference is inactive.

`portsAudioFree`

Type: **integer**

Count of the currently unused audio ports.

Zero if the conference is inactive.

`portsContentFree`

Type: **integer**

Count of the currently unused content ports.

Zero if the conference is inactive.

`numericID`

Type: **string** (up to 80 characters)

Used for registration with H.323 gatekeeper / SIP registrar, and to dial in to the conference.

This is an empty string if the parameter is not set.

`pin`

Type: **string** (up to 40 characters)

The PIN for this conference. If associated with a conference, it is a string of numeric digits that must be entered to gain access to that conference.

Note: A PIN is only valid for incoming calls—no outgoing calls will ever need to enter it. As a result of this, a conference PIN can only be set when the conference has a numeric ID. Trying to set a PIN without a numeric ID will return a fault, and clearing a conference's numeric ID will also clear that conference's PIN.

`registerWithGatekeeper`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the H.323 gatekeeper.

`registerWithSIPRegistrar`

Type: **boolean**

Defines whether or not this item registers its **`numericID`** with the SIP registrar.

`recording`

Type: **boolean**

True if this conference is being recorded by a recording device specified in **`conference.invite`**.

`audioPortLimitSet`

Type: **boolean**

Defines whether the **`audioPortLimit`** is applied.

Value	Description
true	Limits the number of audio ports to the value in **`audioPortLimit`**
false	**`audioPortLimit`** is ignored if it is present

`audioPortLimit`

Type: **integer**

The limit on the number of audio ports this conference may allow.

This may be returned as **`0`**, even though the audio ports are not limited to **`0`**, unless **`audioPortLimitSet`** is **`true`**.

`videoPortLimitSet`

Type: **boolean**

Defines whether the **`videoPortLimit`** is applied.

Value	Description
true	Limits the number of video ports to the value in **`videoPortLimit`**
false	**`videoPortLimit`** is ignored if it is present

`videoPortLimit`

Type: **integer**

The limit on the number of video ports this conference may allow.

This may be returned as **`0`**, even though the video ports are not limited to **`0`**, unless **`videoPortLimitSet`** is **`true`**.

`participantList`

Type: **array**

Array of participants. Each member of the array is a struct that represents a participant on the TelePresence Server.

`participantList` array members

The returned **`participantList`** is an array of the conference's participants. Note that the member structs of this array are different to those returned in **`participantList`** by the **`conference.invite`** call.

If there are no participants in this conference, the **`participantList`** array is returned empty.

Note: it is only returned empty if no participants have joined the conference since the TelePresence Server started up or there are no active participants and the **Clear previous participants record** button on the web interface has been clicked. Otherwise it will list disconnected participants.

Each struct contains the following parameters:

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`participantID`

Type: **integer**

Deprecated. Use **`participantGUID`** instead.

The unique ID of this participant, assigned by the TelePresence Server.

The TelePresence Server returns this parameter, even though it is deprecated, to ensure application compatibility in the short term. We recommend that you use the replacement parameter instead.

`callState`

Type: **integer**

State of the call between the TelePresence Server and this participant.

Value	Description
0	Not connected
1	Calling in (not yet in conference)
2	Called in and participating
3	Calling out (not yet in conference)
4	Called out and participating

`endpointType`

Type: **integer**

Deprecated: use **`endpointCategory`** instead.

Value	Description
1	Normal endpoint
3	Grouped endpoints
4	T3
5	Cisco CTS or other TIP capable endpoints

The TelePresence Server returns this parameter, even though it is deprecated, to ensure application compatibility in the short term. We recommend that you use the replacement parameter instead.

`endpointCategory`

Type: **string**

Value	Description
normal	Normal endpoint
group	Grouped endpoints
t3	T3
cts	Cisco CTS or other TIP capable endpoints

`callStartMute`

Type: **boolean**

True if this endpoint is being sent black video during call setup.

`master`

Type: **boolean**

Value	Description
true	This endpoint is conference master
false	(default if omitted) This endpoint is not the conference master

`callType`

Type: **string**

Value	Description
audio	An audio only participant
video	A video participant

`callProtocol`

Type: **string**

Value	Description
sip	This call uses the SIP protocol.
h323	This call uses the H.323 protocol.

Conditional:

`enumerateID`

Type: **integer**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of results. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

Only returned if there is more data to return than can be contained in one response.

disconnectReason

Type: **string**

The reason why the endpoint disconnected.

Value	Description
unspecified	Unspecified error
localTeardown	Requested by administrator
noAnswer	No answer
rejected	Call rejected
busy	Busy
gatekeeperError	Gatekeeper error
remoteTeardown	Left conference
timeout	Call timed out
protocolError	Protocol error
unreachable	Endpoint is unreachable
networkError	Network error
capabilityNegotiationError	Capability negotiation error
dnsFailure	DNS failure

Only returned for disconnected participants.

rxPreviewURL

Type: **string**

The URL to retrieve a jpeg snapshot of video received from this participant.

Only returned for active participants.

txPreviewURL

Type: **string**

The URL to retrieve a jpeg snapshot of video sent to this participant.

Only returned for active participants.

callDuration

Type: **integer**

The duration of the call in seconds.

Only returned for active participants.

callDirection

Type: **string**

This parameter is not present if `callState` is `0` (not connected).

Value	Description
incoming	The participant called in to the TelePresence Server
outgoing	The TelePresence Server called out to the participant

Only returned for active participants.

`callBandwidth`

Type: **integer**

Call bandwidth in kbps.

Only returned for active participants.

`micMute`

Type: **boolean**

True if far end microphone is muted.

`recordingDevice`

Type: **boolean**

Value	Description
true	The endpoint is treated as a recording device; it does not feature in the layout and other participants are made aware of its presence by a red dot as appropriate.
false	(default if omitted) The endpoint is a normal endpoint.

Only returned for active participants.

`txAudioMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the audio signal transmitted to this endpoint.

Only returned for active participants.

`rxAudioMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the audio signal received from this endpoint.

Only returned for active participants.

`txVideoMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the video signal transmitted to this endpoint.

Only returned for active participants.

`rxVideoMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the video signal received from this endpoint.

Only returned for active participants.

`isImportant`

Type: **boolean**

Defines whether the participant is important (i.e. the participant's transmitted video is given preference over others when composing video).

Value	Description
true	The participant is important
false	(Default if omitted) The participant's video is not given preference over other that of the other participants

Only returned for active participants.

conference.uninvite

Status: **active**

Removes participants from the specified conference. This call requires one conference identification parameter and one participant list parameter.

The call returns a fault if it cannot find a specified participant, even if the TelePresence Server has successfully uninvited the other specified participants.

Accepts:

Required:

To identify the conference, use `conferenceGUID` instead of `conferenceID`, not both.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use `conferenceGUID` instead.

Unique conference identifier.

Optional:

To identify participants to uninvite, **use only one of the following** optional parameters.

`participantListGUID`

Type: **string**

Comma separated list of `participantGUID`s that identifies which participants to remove from this conference. For example, `C8200C3F-49CE-4763-98E0-790B4F038995, B1101410-6BB8-487E-9D6F-91E810E80651`.

`participantList`

Type: **string**

A comma separated list of participant addresses.

Value	Description
Example string	`10.2.171.232, 10.47.2.246, h323:numericID@domain.com`

`participantListID`

Type: **string**

Deprecated. Use `participantListGUID` instead.

Comma separated list of `participantID`s that identifies which participants to remove from the conference. For example; `1024, 1056`.

device.query

Status: **active**

Returns high level status information about the device. It does not accept parameters other than the authentication strings.

Returns:

`currentTime`

Type: **dateTime.iso8601**

The system's current time (UTC).

`restartTime`

Type: **dateTime.iso8601**

The date and time when the system was last restarted.

`uptime`

Type: **integer**

The number of seconds since the last restart.

`serial`

Type: **string**

The serial number of the device.

`apiVersion`

Type: **string**

The version number of the API implemented by this device.

`activatedFeatures`

Type: **array**

Each member contains a string named **`feature`** containing a short description of that feature, for example, **`Encryption`**.

`shutdownStatus`

Type: **string**

Displays one of the following: **`notShutdown`**, **`shutdownInProgress`**, **`shutdown`** or **`error`**.

device.network.query

Status: **active**

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.

Returns:

`dns`

Type: **struct**

An array whose members represent the device's DNS parameters.

`portA`

Type: **struct**

A structure that contains configuration and status information for Ethernet port A on the device.

`dns` array

`hostName`

Type: **string**

The host name of queried device.

`nameServer`

Type: **string**

The IP address of the name server, in dotted quad format (IPv4) or CIDR format (IPv6).

`nameServerSecondary`

Type: **string**

The IP address of the secondary name server, in dotted quad format (IPv4) or CIDR format (IPv6).

`domainName`

Type: **string**

The domain name (DNS suffix).

Port arrays structure

`enabled`

Type: **boolean**

Value	Description
true	The subject of this call or response is enabled.
false	The subject of this call or response is not enabled.

`ipv4Enabled`

Type: **boolean**

`true` if IPv4 interface is enabled.

`ipv6Enabled`

Type: **boolean**

`true` if IPv6 interface is enabled.

`linkStatus`

Type: **boolean**

Value	Description
true	The ethernet connection to this port is active.
false	The ethernet connection to this port is not active.

`speed`

Type: **integer**

Speed of the connection on this Ethernet interface. One of 10, 100 or 1000, in Mbps.

`fullDuplex`

Type: **boolean**

Value	Description
true	The port can support a full-duplex connection.
false	The port can support a half-duplex connection.

`macAddress`

Type: **string**

The MAC address of this interface. A 12 character string of hex digits with no separators.

`packetsSent`

Type: **integer**

The number of packets sent from this Ethernet port. Note that this value is a 32-bit signed integer, and thus may wrap.

`packetsReceived`

Type: **integer**

The number of packets received on this Ethernet port. Note that this value is a 32-bit signed integer, and thus may wrap.

`multicastPacketsSent`

Type: **integer**

Number of multicast packets sent from this Ethernet interface. Note that this value is a 32-bit signed integer, and thus may wrap.

`multicastPacketsReceived`

Type: **integer**

Number of multicast packets received on this Ethernet interface. Note that this value is a 32-bit signed integer, and thus may wrap.

`bytesSent`

Type: **integer**

The number of bytes sent by the device. Note that this value is a 32-bit signed integer, and thus may wrap.

`bytesReceived`

Type: **integer**

The number of bytes received by the device. Note that this value is a 32-bit signed integer, and thus may wrap.

`queueDrops`

Type: **integer**

Number of packets dropped from the queue on this network interface. Note that this value is a 32-bit signed integer, and thus may wrap.

`collisions`

Type: **integer**

Count of the network collisions recorded by the device. Note that this value is a 32-bit signed integer, and thus may wrap.

`transmitErrors`

Type: **integer**

The count of transmission errors on this Ethernet interface. Note that this value is a 32-bit signed integer, and thus may wrap.

`receiveErrors`

Type: **integer**

The count of receive errors on this interface. Note that this value is a 32-bit signed integer, and thus may wrap.

`bytesSent64`

Type: **string**

64 bit versions of the **`bytesSent`** statistic, using a string rather than an integer.

`bytesReceived64`

Type: **string**

64 bit versions of the **`bytesReceived`** statistic, using a string rather than an integer.

Conditional:

The following parameters are returned only if the interface is enabled and configured.

`dhcpv4`

Type: **boolean**

Value	Description
true	The device's IP address is allocated by DHCP
false	The device's IP address is manually configured

`ipv4Address`

Type: **string**

IPv4 IP address in dotted-quad format.

`ipv4SubnetMask`

Type: **string**

The IPv4 subnet mask in dotted quad format.

`defaultipv4Gateway`

Type: **string**

The device's IPv4 default gateway in dotted quad format.

`ipv6Conf`

Type: **string**

Indicates how the IPv6 address is assigned.

Value	Description
automatic	This interface's IPv6 address is configured automatically (by SLAAC/DHCPv6)
manual	This interface's IPv6 address is configured manually

ipv6Address

Type: **string(63)**

The IPv6 address in CIDR format.

ipv6PrefixLength

Type: **integer**

The length of the IPv6 address prefix.

defaultIpv6Gateway

Type: **string(63)**

The address of the IPv6 default gateway in CIDR format.

linkLocalIpv6Address

Type: **string(63)**

The link local IPv6 address in CIDR format.

linkLocalIpv6PrefixLength

Type: **integer**

Length of the link local IPv6 address prefix.

device.health.query

Status: **active**

Returns the current status of the device, such as health monitors and CPU load.

Returns:

cpuLoad

Type: **integer**

The CPU load as a percentage of the maximum.

fanStatus

Type: **string**

Value	Description
ok	The fan is currently functioning properly.
outOfSpec	The fan is currently not functioning properly.

fanStatus is only returned on 7010 devices, not 8710s.

fanStatusWorst

Type: **string**

Value	Description
ok	The fan has been functioning properly since the device was booted.
outOfSpec	The fan has not been functioning properly since the device was booted.

fanStatusWorst is only returned on 7010 devices, not 8710s.

temperatureStatus

Type: **string**

Value	Description
ok	The temperature is currently within the normal operating range.
outOfSpec	The temperature is currently outside the normal operating range.
critical	The temperature is too high and the device will shutdown if this condition persists.

temperatureStatusWorst

Type: **string**

The worst temperature status recorded on this device since it booted.

Value	Description
ok	The temperature has been within the normal operating range since the device was booted.
outOfSpec	The temperature has been outside the normal operating range at least once since the device was booted.
critical	At some point since the last boot the temperature was too high. The device will shutdown if this condition persists.

rtcBatteryStatus

Type: **string**

The current status of the RTC battery (Real Time Clock).

Value	Description
ok	The battery is operating within the normal range.
outOfSpec	The battery is operating outside of the normal range, and may require service.

rtcBatteryStatusWorst

Type: **string**

The worst recorded status of the RTC battery.

Value	Description
ok	The battery has been operating inside the normal range since the device was booted.
outOfSpec	The battery has operated outside of the normal range at some time since the device was booted.

`voltagesStatus`

Type: **string**

Value	Description
ok	The voltage is currently within the normal range
outOfSpec	The voltage is currently outside the normal range

`voltagesStatusWorst`

Type: **string**

Value	Description
ok	The voltage has been within the normal range since the device last booted.
outOfSpec	The voltage has been outside the normal range at some time since the device last booted

`operationalStatus`

Type: **string**

Value	Description
active	The TelePresence Server is acive.
shuttingDown	The device is shutting down.
shutDown	The device has shut down.
unknown	The operational status of the device is unknown.

device.restartlog.query

Status: **active**

Returns the restart log - also known as the system log on the web interface.

Returns:

`log`

Type: **array**

Each member of the array contains log information (called system log in the user interface).

`log` array members

`time`

Type: **dateTime.iso8601**

The time when the device restarted.

Value	Description
20110119T13:52:42	yyyymmddThh:mm:ss

`reason`

Type: **string**

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

device.restart

Status: **active**

Restarts the device, or shuts it down without a restart.

Accepts:

Optional:

`shutdownOnly`

Type: **boolean**

Value	Description
true	The device will shut down when it receives this request and will not restart.
false	This is the default value.

participant.diagnostics

Status: **active**

The call specifies which participant's diagnostics to retrieve and also a listening interface for the returned information.

The reason for providing `receiverURI` is because the call is asynchronous; you should receive an "Operation successful" result slightly before the data returns (an XML-RPC methodCall with methodName `participantDiagnosticsResponse`) on the listening interface.

This method may return `Fault 203: 'too many asynchronous requests` this means that the TelePresence Server is currently processing the maxmimum possible number of diagnostics requests and another request cannot be processed at this time. The request should be retried later.

The returned information contains arrays comprising the different types of data streams between this participant and the hosting TelePresence Server. Each array member represents a single stream.

Example XML-RPC response to participant.diagnostics [p.154]

If there are no streams of a particular type, the corresponding array is returned empty.

Accepts:

Required:

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`receiverURI`

Type: **string** (up to 255 characters)

Fully-qualified URI that identifies the listening application's XML-RPC interface (protocol, address, and port), for example, `http://tms1:8080/RPC2`. You can use `http` or `https` and, if no port number is specified, the device will use the protocol defaults (80 and 443 respectively).

Optional:

`sourceIdentifier`

Type: **string** (up to 255 characters—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.

Returns:

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`sourceIdentifier`

Type: **string** (up to 255 characters—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.

`audioRx`

Type: **array**

An array of structs, each of which represents an audio stream received from the participant's endpoint.

`audioTx`

Type: **array**

An array of structs, each of which represents an audio stream transmitted to the participant's endpoint.

`auxiliaryAudioRx`

Type: **array**

An array of structs, each of which represents an auxiliary audio stream received from the participant's endpoint.

`auxiliaryAudioTx`

Type: **array**

An array of structs, each of which represents an auxiliary audio stream transmitted to the participant's endpoint.

`videoRx`

Type: **array**

An array of structs, each of which represents a video stream received from the participant's endpoint.

`videoTx`

Type: **array**

An array of structs, each of which represents a video stream transmitted to the participant's endpoint.

`contentVideoRx`

Type: **array**

An array of structs, each of which represents a content video stream received from the participant's endpoint.

`contentVideoTx`

Type: **array**

An array of structs, each of which represents a content video stream transmitted to the participant's endpoint.

Contents of diagnostics arrays:

Each of the above arrays may contains zero or more stream structs. Each stream struct will contain relevant parameter/value pairs from the following lists:

All streams:

`codec`

Type: **string**

The codec in use, or `other` for undefined codecs.

`encrypted`

Type: **boolean**

True if the stream data is encrypted.

`muted`

Type: **boolean**

True if the stream is muted.

`channelBitRate`

Type: **integer**

Bit rate of the channel in bits per second (bps).

Conditional, depending on the type and direction of the stream:

`packetsSent`

Type: **integer**

Count of packets sent in this stream.

`packetsReceived`

Type: **integer**

Count of packets received in this stream.

`packetErrors`

Type: **integer**

Count of packets with errors in this stream.

`packetsMissing`

Type: **integer**

Count of packets missing from this stream.

`framesReceived`

Type: **integer**

Count of frames received in this stream.

`frameErrors`

Type: **integer**

Count of frames with errors in this stream.

`jitter`

Type: **integer**

Current jitter in this stream, measured in milliseconds (ms).

`energy`

Type: **integer**

The level of the signal, supplied in decibels (dB).

`configuredBitRate`

Type: **integer**

The configured bit rate of this stream, in bits per second (bps).

`configuredBitRateReason`

Type: **string**

Value	Description
aggregateBandwidth	The TelePresence Server has limited the bit rate so that multiple streams can be sent without exceeding a given limit on overall bandwidth.
flowControl	The far end has requested that the TelePresence Server sends video at a lower bit rate.
notLimited	The configured bit rate is not limited by **`flowControl`** or **`aggregateBandwidth`**.

`expectedBitRate`

Type: **integer**

The expected bit rate of this stream, in bits per second (bps).

`expectedBitRateReason`

Type: **string**

Value	Description
viewedSize	The TelePresence Server requested a reduction in the bitrate of the video stream because the video stream from that endpoint is not being displayed at full size.
errorPackets	The TelePresence Server requested a reduction in the bitrate of the video stream because there are errors in the video stream.
notLimited	The TelePresence Server has not requested a reduction in the bitrate of the video stream.

`actualBitRate`

Type: **integer**

The measured bit rate of this stream, in bits per second (bps).

`frameRate`

Type: **integer**

The frame rate of the video stream, in frames per second (fps).

`fastUpdateRequestsSent`

Type: **integer**

The count of fast update requests sent in this stream.

`fastUpdateRequestsReceived`

Type: **integer**

The count of fast update requests received in this stream.

participant.enumerate

Status: **active**

Returns an array of the participants who are active on the queried TelePresence Server. Endpoints that are either connecting or inactive at the time of the enumeration are not included in the response.

Accepts:

Optional:

`enumerateID`

Type: **integer**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of results. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

Returns:

If there are no participants to enumerate, then the **`participant.enumerate`** call does not return the **`participants`** array.

Conditional:

`participants`

Type: **array**

An array of structures that represent participants.

`enumerateID`

Type: **integer**

Enumerate calls may return many results so all of them will accept this parameter and may include this parameter in the response.

If the response includes an enumerateID, the application should pass the ID to the subsequent enumerate call to retrieve the next set of results. If the response does not include an enumerateID, there are no more results in the enumeration.

If the application omits the enumerateID, the target device will start a new enumeration and return the first set of results.

`participants` array

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`participantID`

Type: **integer**

Deprecated. Use **`participantGUID`** instead.

The unique ID of this participant, assigned by the TelePresence Server.

`conferenceGUID`

Type: **string**

Globally unique identifier of the conference.

`conferenceID`

Type: **integer**

Deprecated. Use **`conferenceGUID`** instead.

Unique conference identifier.

`address`

Type: **string**

The address of the item, e.g. endpoint or gateway; may be hostname, IP address or E.164 number.

`endpointCategory`

Type: **string**

Value	Description
normal	Normal endpoint
group	Grouped endpoints
t3	T3
cts	Cisco CTS or other TIP capable endpoints

`callProtocol`

Type: **string**

Value	Description
sip	This call uses the SIP protocol.
h323	This call uses the H.323 protocol.

`callState`

Type: **integer**

State of the call between the TelePresence Server and this participant.

Value	Description
0	Not connected
1	Calling in (not yet in conference)
2	Called in and participating
3	Calling out (not yet in conference)
4	Called out and participating

participant.set

Status: **active**

Changes the state of the supplied parameters for the specified participant.

Accepts:

Required:

To identify the participant, use **`participantGUID`** instead of **`participantID`**, not both.

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`participantID`

Type: **integer**

Deprecated. Use `participantGUID` instead.

The unique ID of this participant, assigned by the TelePresence Server.

Optional:

`txAudioMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the audio signal transmitted to this endpoint.

`rxAudioMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the audio signal received from this endpoint.

`txVideoMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the video signal transmitted to this endpoint.

`rxVideoMute`

Type: **boolean**

Defines whether the TelePresence Server mutes the video signal received from this endpoint.

`isImportant`

Type: **boolean**

Defines whether the participant is important (i.e. the participant's transmitted video is given preference over others when composing video).

Value	Description
true	The participant is important
false	(Default if omitted) The participant's video is not given preference over other that of the other participants

participant.tidylayout

Status: **active**

Tidies up the composed video layout sent to the specified participant's endpoint.

Accepts:

Required:

`participantGUID`

Type: **string**

The GUID of this participant, assigned by the TelePresence Server.

`participantID`

Type: **integer**

Deprecated. Use `participantGUID` instead.

The unique ID of this participant, assigned by the TelePresence Server.

system.info

Status: **active**

Returns the current status of the queried system.

Returns:

`gateKeeperOK`

Type: **boolean**

True if the gatekeeper is configured and the TelePresence Server is registered.

`makeCallsOK`

Type: **boolean**

True if the system has enough resources to make at least one call.

`tpsNumberOK`

Type: **integer**

The count of configured and active TelePresence Servers.

`tpdVersion`

Type: **string**

The TelePresence Server software version number.

`tpdName`

Type: **string**

The TelePresence Server system name.

`tpdUptime`

Type: **integer**

The period of time (in seconds) that has passed since the system booted.

`tpdSerial`

Type: **string**

The serial number of the TelePresence Server.

`portsVideoTotal`

Type: **integer**

The total number of video ports.

`portsVideoFree`

Type: **integer**

Count of the currently unused video ports.

`portsAudioTotal`

Type: **integer**

The total number of audio ports.

`portsAudioFree`

Type: **integer**

Count of the currently unused audio ports.

`portsContentTotal`

Type: **integer**

The total number of content ports.

`portsContentFree`

Type: **integer**

Count of the currently unused content ports.

`maxConferenceSizeVideo`

Type: **integer**

The count of unused video ports on the least-used TelePresence Server controlled by this unit. Indicates the maximum number of video ports that could currently be allocated to a single conference.

`maxConferenceSizeAudio`

Type: **integer**

The count of unused audio-only ports on the least-used TelePresence Server controlled by this unit. Indicates the maximum number of audio-only ports that could currently be allocated to a single conference.

`maxConferenceSizeContent`

Type: **integer**

The count of unused content ports on the least-used TelePresence Server controlled by this unit. Indicates the maximum number of content ports that could currently be allocated to a single conference.

`numControlledServers`

Type: **integer**

The number of TelePresence Servers controlled by this unit (including itself).

`operationMode`

Type: **string**

The operation mode is one of **standalone** (locally managed), **flexible** (remotely managed) or **slave** (backplane slave blade).

`licenseMode`

Type: **string**

The license mode is one of the following:.

If **operationMode** = **standalone**: one of **HD** or **fullHD**.

If **operationMode** = **flexible**: always flexible.

If **operationMode**= **slave**: not present.

feedbackReceiver.query

Status: **active**

This call asks the device for a list of all the feedback receivers that have previously been configured. It does not accept parameters other than the authentication strings.

Returns:

If there are no feedback receivers to enumerate, then the **`feedbackReceiver.query`** returns an empty **`receivers`** array.

`receivers`

Type: **array**

An array of feedback receivers, with members corresponding to the entries in the receivers table on the device's web interface.

`receivers` array members

Each receiver in the response contains the following parameters:

`receiverURI`

Type: **string** (up to 255 characters)

Fully-qualified URI that identifies the listening application's XML-RPC interface (protocol, address, and port), for example, **`http://tms1:8080/RPC2`**. You can use **`http`** or **`https`** and, if no port number is specified, the device will use the protocol defaults (80 and 443 respectively).

`sourceIdentifier`

Type: **string** (up to 255 characters—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.

`index`

Type: **integer**

A number that identifies the position of the item in context with similar items.

The **`index`** describes the position of this feedback receiver in the TelePresence Server's table of feedback receivers. It is a number between 1 and 20 (inclusive).

feedbackReceiver.configure

Status: **active**

This call configures the device to send feedback about the specified **`subscribedEvents`** to the specified **`receiverURI`**.

Accepts:

Required:

`receiverURI`

Type: **string** (up to 255 characters)

Fully-qualified URI that identifies the listening application's XML-RPC interface (protocol, address, and port), for example, **`http://tms1:8080/RPC2`**. You can use **`http`** or **`https`** and, if no port number is specified, the device will use the protocol defaults (80 and 443 respectively).

Optional:

`sourceIdentifier`

Type: **string** (up to 255 characters—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.

`receiverIndex`

Type: **integer**

The position of this feedback receiver in the device's table of feedback receivers.

Value	Description
-1	The feedback receiver will use any available position.
1	The first position in the table (this value is assumed if you don't supply **`receiverIndex`** - *overwriting any existing entry in position 1*)
20	The 20th (maximum allowed) position

We recommend that you set **`receiverIndex`** to **`-1`** in this call. This ensures that the TelePresence Server allocates an available slot and that you don't inadvertently overwrite an existing feedback receiver in slot 1.

`subscribedEvents`

Type: **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.

You may specify any or all of the following:

- cdrAdded
- conferenceStarted
- conferenceFinished
- conferenceActive
- conferenceInactive
- configureAck
- deviceStatusChanged
- participantJoined
- participantLeft
- participantConnected
- participantDisconnected
- receiverModified
- receiverDeleted
- restart

If this array is absent, the receiver subscribes to all notifications by default.

Returns:

The call returns the allocated **`receiverIndex`**.

`receiverIndex`

Type: **integer**

The position of this feedback receiver in the device's table of feedback receivers.

feedbackReceiver.reconfigure

Status: **active**

This call reconfigures an existing feedback receiver.This call only reconfigures the receiver parameters that you specify; the TelePresence Server retains the original values for any parameters that you omit.

Accepts:

Required:

`receiverIndex`

Type: **integer**

The position of this feedback receiver in the device's table of feedback receivers.

The call returns a fault if there is no feedback receiver at the specified **`receiverIndex`**.

Optional:

`receiverURI`

Type: **string** (up to 255 characters)

Fully-qualified URI that identifies the listening application's XML-RPC interface (protocol, address, and port), for example, **`http://tms1:8080/RPC2`**. You can use **`http`** or **`https`** and, if no port number is specified, the device will use the protocol defaults (80 and 443 respectively).

The call returns a fault if you supply an empty **`receiverURI`**. However, if you omit the parameter altogether, the original value persists.

`sourceIdentifier`

Type: **string** (up to 255 characters—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`

Type: **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.

You may specify any or all of the following:

- cdrAdded
- conferenceStarted
- conferenceFinished
- conferenceActive
- conferenceInactive
- configureAck
- deviceStatusChanged
- participantJoined
- participantLeft

- participantConnected
- participantDisconnected
- receiverModified
- receiverDeleted
- restart

If this array is absent, the receiver's existing subscriptions will not be changed from the values created by the original `feedbackReceiver.configure` call.

feedbackReceiver.remove

Status: **active**

Removes the specified feedback receiver.

Accepts:

Required:

`receiverIndex`

Type: **integer**

The position of this feedback receiver in the device's table of feedback receivers.

The call returns a fault if there is no feedback receiver at the specified `receiverIndex`.

feedbackReceiver.status

Status: **active**

This call asks the device for a list of all the events that a feedback receiver is subscribed to.

Accepts:

Required:

`receiverIndex`

Type: **integer**

The position of this feedback receiver in the device's table of feedback receivers.

The call returns a fault if there is no feedback receiver at the specified `receiverIndex`.

Returns:

If there are no events subscribed to by the feedback receivers to enumerate, then the `feedbackReceiver.status` returns an empty `receivers` array.

`receiverIndex`

Type: **integer**

The position of this feedback receiver in the device's table of feedback receivers.

`receiverURI`

Type: **string** (up to 255 characters)

Fully-qualified URI that identifies the listening application's XML-RPC interface (protocol, address, and port), for example, **`http://tms1:8080/RPC2`**. You can use **`http`** or **`https`** and, if no port number is specified, the device will use the protocol defaults (80 and 443 respectively).

`sourceIdentifier`

Type: **string** (up to 255 characters—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`

Type: **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.

You may specify any or all of the following:

- cdrAdded
- conferenceStarted
- conferenceFinished
- conferenceActive
- conferenceInactive
- configureAck
- deviceStatusChanged
- participantJoined
- participantLeft
- participantConnected
- participantDisconnected
- receiverModified
- receiverDeleted
- restart

Related information

system.xml file

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>
    <serial>SM021037</serial>
    <softwareVersion>2.3(1.48)</softwareVersion>
    <buildVersion>13.0(1.48)</buildVersion>
    <hostName>dt12b7</hostName>
    <ipAddress>10.47.223.127</ipAddress>
    <ipAddressV6>2001:420:40ff:ff0a:20d:7cff:fe10:81b8</ipAddressV6>
    <macAddress>00:0D:7C:10:81:B8</macAddress>
    <gatekeeperUsage>Yes</gatekeeperUsage>
    <gatekeeperAddress>mainvcs.test.lal</gatekeeperAddress>
    <gatekeeperIds>dt12b7,dt12b7-l,dt12b7-c,dt12b7-r</gatekeeperIds>
    <sipRegistrarUsage>Yes</sipRegistrarUsage>
    <sipRegistrarAddressmainvcs.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>
```

System XML contents

Node name	Node contents
gatekeeperUsage	**`Yes`**: gatekeeper usage is enabled **`No`**: gatekeeper usage is disabled
gatekeeperAddress	The gatekeeper 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
sipRegistrarAddress	The SIP registrar IP address.
sipRegistrarDomain	The SIP registrar domain name.

Node name	Node contents
sipTrunkUsage	**Yes**: trunk usage is enabled **No**: trunk usage is disabled
sipTrunkAddress	The SIP trunk IP address.
sipTrunkDomain	The SIP trunk domain name.
isMaster	Determines whether this system controls its own calls and conferences. This will be either: **Yes**: when clusterType is **unclustered** or **master** **No**: when clusterType is **slave**
clusterType	The role of this system in a backplane cluster. This will be one of the following: **unclustered**, **master**, or **slave**
totalVideoPorts	The total number of video ports supported by the device.
totalContentPorts	The total number of content ports supported by the device.
totalAudioOnlyPorts	The total number of audio only ports supported by the device.
uptimeSeconds	The length of time in seconds that the device has been up and running.

Example XML-RPC response to `participant.diagnostics`

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
  <methodName>participantDiagnosticsResponse</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>participantGUID</name>
            <value>
              <string>1f983510-8843-11e0-bd5d-000d7c005ce8</string>
            </value>
          </member>
          <member>
            <name>sourceIdentifier</name>
            <value>
              <string>UserPC</string>
            </value>
          </member>
          <member>
            <name>audioRx</name>
            <value>
              <array>
                <data>
                  <value>
                    <struct>
                      <name>codec</name>
                      <value>
                        <string>G.722</string>
                      </value>
                      <name>encrypted</name>
                      <value>
                        <boolean>1</boolean>
                      </value>
                      <name>channelBitRate</name>
                      <value>
                        <int>64000</int>
                      </value>
                      <name>jitter</name>
                      <value>
                        <int>0</int>
                      </value>
                      <name>energy</name>
                      <value>
                        <int>-34</int>
                      </value>
                      <name>packetsReceived</name>
                      <value>
                        <int>6951</int>
                      </value>
                      <name>packetErrors</name>
                      <value>
                        <int>0</int>
                      </value>
                      <name>packetsMissing</name>
                      <value>
                        <int>0</int>
                      </value>
                      <name>framesReceived</name>
                      <value>
                        <int>277960</int>
                      </value>
                      <name>frameErrors</name>
                      <value>
                        <int>0</int>
                      </value>
                      <name>muted</name>
                      <value>
                        <boolean>0</boolean>
                      </value>
                    </struct>
                  </value>
                </data>
              </array>
            </value>
          </member>
          <member>
            <name>audioTx</name>
            <value>
              <array>
                <data>
                  <value>
                    <struct>
                      <name>codec</name>
                      <value>
                        <string>G.722</string>
                      </value>
                      <name>encrypted</name>
                      <value>
                        <boolean>1</boolean>
                      </value>
                      <name>channelBitRate</name>
                      <value>
```

```
              <int>64000</int>
            </value>
            <name>packetsSent</name>
            <value>
              <int>6994</int>
            </value>
            <name>muted</name>
            <value>
              <boolean>0</boolean>
            </value>
          </struct>
        </value>
      </data>
    </array>
  </value>
</member>
<member>
  <name>auxiliaryAudioRx</name>
  <value>
    <array>
      <data/>
    </array>
  </value>
</member>
<member>
  <name>auxiliaryAudioTx</name>
  <value>
    <array>
      <data/>
    </array>
  </value>
</member>
<member>
  <name>videoRx</name>
  <value>
    <array>
      <data>
        <value>
          <struct>
            <name>codec</name>
            <value>
              <string>H.264</string>
            </value>
            <name>height</name>
            <value>
              <int>0</int>
            </value>
            <name>width</name>
            <value>
              <int>0</int>
            </value>
            <name>encrypted</name>
            <value>
              <boolean>1</boolean>
            </value>
            <name>channelBitRate</name>
            <value>
              <int>448000</int>
            </value>
```

```
              <name>expectedBitRate</name>
              <value>
                <int>256000</int>
              </value>
              <name>expectedBitRateReason</name>
              <value>
                <string>viewedSize</string>
              </value>
              <name>actualBitRate</name>
              <value>
                <int>0</int>
              </value>
              <name>jitter</name>
              <value>
                <int>0</int>
              </value>
              <name>packetsReceived</name>
              <value>
                <int>0</int>
              </value>
              <name>packetErrors</name>
              <value>
                <int>0</int>
              </value>
              <name>framesReceived</name>
              <value>
                <int>0</int>
              </value>
              <name>frameErrors</name>
              <value>
                <int>0</int>
              </value>
              <name>frameRate</name>
              <value>
                <int>0</int>
              </value>
              <name>fastUpdateRequestsSent</name>
              <value>
                <int>0</int>
              </value>
              <name>muted</name>
              <value>
                <boolean>0</boolean>
              </value>
            </struct>
          </value>
        </data>
      </array>
    </value>
  </member>
  <member>
    <name>videoTx</name>
    <value>
      <array>
        <data>
          <value>
            <struct>
              <name>codec</name>
              <value>
```

```
                  <string>H.263+</string>
                </value>
                <name>height</name>
                <value>
                  <int>576</int>
                </value>
                <name>width</name>
                <value>
                  <int>704</int>
                </value>
                <name>encrypted</name>
                <value>
                  <boolean>1</boolean>
                </value>
                <name>channelBitRate</name>
                <value>
                  <int>448000</int>
                </value>
                <name>configuredBitRate</name>
                <value>
                  <int>448000</int>
                </value>
                <name>configuredBitRateReason</name>
                <value>
                  <string>notLimited</string>
                </value>
                <name>actualBitRate</name>
                <value>
                  <int>47468</int>
                </value>
                <name>packetsSent</name>
                <value>
                  <int>2054</int>
                </value>
                <name>frameRate</name>
                <value>
                  <int>15</int>
                </value>
                <name>fastUpdateRequestsReceived</name>
                <value>
                  <int>2</int>
                </value>
                <name>muted</name>
                <value>
                  <boolean>0</boolean>
                </value>
              </struct>
            </value>
          </data>
        </array>
      </value>
    </member>
    <member>
      <name>contentVideoRx</name>
      <value>
        <array>
          <data/>
        </array>
      </value>
```

```
          </member>
          <member>
            <name>contentVideoTx</name>
            <value>
              <array>
                <data/>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

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.

Fault Code	Description
1	**`Method not supported`**. This method is not supported on this device or is unknown.
2	**`Duplicate conference name`**. A conference name was specified, but is already in use.
4	**`No such conference or auto attendant`**. The conference or auto attendant identification given does not match any conference or auto attendant.
5	**`No such participant`**. The participant identification given does not match any participants.
6	**`Too many conferences`**. The device has reached the limit of the number of conferences that can be configured.
8	**`No conference name or auto attendant id supplied`**. A conference name or auto attendant identifier was required, but was not present.
10	**`No participant address supplied`**. A participant address is required but was not present.
13	**`Invalid PIN specified`**. A PIN specified is not a valid series of digits.
15	**`Insufficient privileges`**. The specified user id and password combination is not valid for the attempted operation.
16	**`Invalid enumerateID value`**. An enumerate ID passed to an enumerate method invocation was invalid. Only values returned by the device should be used in enumerate methods.
17	**`Port reservation failure`**. There are insufficient free ports to complete/place the requested calls.
18	**`Duplicate numeric ID`**. A numeric ID was given, but this ID 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	**`New port limit lower than currently active`**
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. Strings are usually limited to 32 characters but may be longer in some cases.
101	**`Missing parameter`**. This is given when a required parameter is absent.
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 a protocol contains an invalid protocol.
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.
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.

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.

Checking for updates and getting help

If you experience any problems when configuring or using the product, consult the online help available from the user interface. The online help explains how the individual features and settings work.

If you cannot find the answer you need, check the web site at http://www.cisco.com/cisco/web/support/index.html where you will be able to:

- make sure that you are running the most up-to-date software,
- find further relevant documentation, for example product user guides, printable versions of the online help, reference guides, and articles that cover many frequently asked questions,
- get help from the Cisco Technical Support team. Make sure you have the following information ready before raising a case:
  - the serial number and product model number of the unit (if applicable)
  - the software build number which can be found on the product user interface (if applicable)
  - your contact email address or telephone number
  - a full description of the problem

References

1. XML-RPC specification (Dave Winer, June 1999); http://www.xmlrpc.com/spec, accessed 24/01/2011.
2. HTTP/1.1 specification (RFC 2616, Fielding et al., June 1999); http://www.ietf.org/rfc/rfc2616.txt, accessed 24/01/2011.
3. Cisco TelePresence CDR logs reference guide (Cisco Systems Inc., Sept 2012); http://www.cisco.com/en/US/docs/telepresence/infrastructure/mcu/admin_guide/cisco_telepresence_infrastructure_cdr_reference_guide.pdf