Cisco TelePresence Conductor

Product Programming Reference Guide

XC2.3

D14948.04

Updated July 2015

Contents

Introduction

This document describes the set of application programming interfaces (APIs) that is available on the Cisco TelePresence Conductor version XC2.3.

There are plans to update the APIs in future releases to better provide a generic API independent of the conference bridge or other resources that the TelePresence Conductor may be managing. These documented APIs are available in version XC2.3, but may change in future releases.

Many of the XC2.3 APIs have a large focus on the Cisco TelePresence MCU. If the conference bridge handling the conference is a Cisco TelePresence Server, the amount of command / status data supported may be less than indicated.

Current TelePresence Conductor API clients

Current clients of the TelePresence Conductor API include:

- Cisco TelePresence Management Suite (Cisco TMS Provisioning Extension, conference control center and scheduled calls)
- Cisco Unified Communications Manager (ad hoc conferencing)
- Prime Collaboration Manager

Remote Management XML-RPC API

The TelePresence Conductor XML-RPC API is loosely based on the existing Cisco TelePresence MCU remote management API.

Some calls are directly processed by the TelePresence Conductor, other calls are composed into messages suitable for underlying TelePresence MCUs or TelePresence Servers and forwarded to the appropriate conference bridge.

Your application should send XML-RPC HTTP POST messages to the URL defined by path **`/RPC2`** on the TelePresence Conductor's IP address, for example **`https://<IP address of the Conductor>/RPC2`**.

The following API calls are supported:

- conference.destroy
- conference.enumerate
- conference.modify
- device.network.query
- device.query
- factory.conferencecreate
- factory.health.query
- factory.webex.add
- feedbackReceiver.extended.configure
- feedbackReceiver.extended.query
- feedbackReceiver.extended.remove
- participant.add
- participant.diagnostics
- participant.disconnect
- participant.enumerate
- participant.message
- participant.modify

Authentication

Note that systems which use XML-RPC over HTTP send authentication over plain text. We therefore recommend using HTTPS instead of HTTP wherever possible.

The controlling application must authenticate itself to the TelePresence Conductor. Also, because the interface is stateless, every call must contain the authentication parameters.

Parameter	Type	Description
`authenticationUser`	string	Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.
`authenticationPassword`	string	The password that corresponds with the given **`authenticationUser`**. The password is case sensitive.

We recommend setting up an administrator account which supports API access and not web access.

XML-RPC return values

The current TelePresence Conductor API is based on the Cisco TelePresence MCU API. Many XML-RPC methods invoked on the TelePresence Conductor are simply proxied to the appropriate TelePresence MCU (s) or TelePresence Server(s) and the responses from the conference bridges are aggregated (if required) and then returned to the client of the TelePresence Conductor API with minimal modification.

The API may respond to requests with empty data structures when the data is not available. Your application must check whether the response includes the expected information, and if not, gracefully handle that situation.

conference.destroy

This call disconnects all participants in the conference and ends the conference. It is used to end ad hoc conferences.

Input parameters

Required parameters

Parameters	Type	Description
`conferenceName`	string	The name of the conference that should end.

Optional parameters

None

Returned data

Parameters	Type	Description
`status`	string	On success: "operation successful".

conference.enumerate

This call returns all active conferences. It works by sending **`conference.enumerate`** messages to all conference bridges configured on the TelePresence Conductor and amalgamating their responses. Only conference data for primary conferences are returned. Cascade specific data will not be returned, however the returned conference struct will be flagged as a cascaded conference.

This call may be resource intensive and may result in multiple messages being sent to the conference bridges.

Input parameters

Required parameters

None

Optional parameters

None

Returned data

Parameters	Type	Description
conferences	array	Array of conference structs as defined by the TelePresence MCU API. TelePresence Conductor will add the following parameters to the structs:
→ **factoryTemplateType**	string	The type of template: *meet*, *lesson* or *unknown*. The type is *unknown*, if the configuration has changed since this conference was started.
→ **isCascaded**	boolean	*true* if the conference is cascaded, *false* otherwise.
→ **factoryConferenceId**	string	Internal conference id used by the TelePresence Conductor.
→ **factoryWebEx**	string	*None* represents a conference that does not support WebEx. (This is the default value.) *SIP* represents a conference that supports a SIP WebEx conference, where resources for one call is used on the primary conference bridge. *SIP-TSP* represents a conference that supports a SIP-TSP WebEx conference, where resources for two calls (one for video, one for audio) are used on the primary conference bridge.
→ **encryption**	string	The encryption requirements for endpoints in this conference. *optional*: endpoints in this conference may have encryption enabled or disabled. (This is the default value.) *required*: endpoints in this conference must have encryption enabled. *forbidden*: endpoints in this conference must not have encryption enabled. This is only applicable to TelePresence Servers, not to TelePresence MCUs.
Other parameters as documented in *Cisco TelePresence MCU API reference guide*. Note that **encryptionRequired** is not supported.		

conference.modify

This call modifies the settings of an existing conference.

Input parameters

Required inputs

Parameters	Type	Description
conferenceName	string	The name of the conference, used to identify which conference to modify.

Optional inputs

Parameters	Type	Description
`h239Important`	boolean	Whether the H.239 channel is set to be important.
`customLayoutEnabled`	boolean	Whether the custom layout has been enabled or not.
`newParticipantsCustomLayout`	boolean	Whether new participants will use the custom layout or not. This is only valid if `customerLayoutEnabled` is *true*.
`customLayout`	integer	The index of the video layout seen by the participant(s). See the relevant *Cisco TelePresence MCU API reference guide* for a list of available layouts and corresponding index values.

Returned data

Parameters	Type	Description
`status`	string	On success: "operation successful".

device.network.query

This call returns network information about the device.

Input parameters

Required inputs

None

Optional inputs

None

Returned data

Parameters	Type	Description
`portA`	struct	A structure that contains configuration information for Ethernet port A on the device.
→ `macAddress`	string	Returns the MAC address of the device.

device.query

This call returns high level status information about the TelePresence Conductor.

Input parameters

Required inputs

None

Optional inputs

None

Returned data

Parameters	Type	Description
`currentTime`	dateTime.iso8601	The system's local current time.
`serial`	string	The serial number of the device.
`softwareVersion`	string	The version number of the software running on the device.
`buildVersion`	string	The build version of the software running on the device.
`model`	string	Cisco TelePresence Conductor
`apiVersion`	string	The version number of the API implemented by this device.
`totalVideoPorts`	integer	The total number of video ports on the device. This includes ports for disabled TelePresence MCUs, but does not include any ports for TelePresence Servers.

factory.conferencecreate

This call creates a new conference via the TelePresence Conductor.

Input parameters

Required parameters

Parameters	Type	Description
`conferenceAlias`	string	The dial string for the new conference to create.

Optional parameters

Parameters	Type	Description
`factoryMinDurationMinutes`	integer	The minimum time a conference will live even with no participants. If the value is 0 the conference will not be created unless there are any auto-dialed participants associated with the conference template. The default is 5.
`factoryMaxDurationMinutes`	integer	The maximum time a conference will live. This value must be equal to or lower than the value defined for the conference duration limit on the conference template, otherwise there will be an error displayed. The default is 0, which means that the limit specified on the conference template is used.

factoryOverridePIN	string	If present, this is the string of numeric digits that participants with chairperson privileges need to enter to join the conference. It overrides the PIN defined on the conference template. For a meeting-type conference this PIN is the PIN used for meeting participants. The string must be less than 32 characters.
factoryOverrideGuestPIN	string	If present, this is the string of numeric digits that participants with guest privileges need to enter to join the conference. It overrides the guest PIN defined on the conference template. The string must be less than 32 characters.
webEx	string	*None* creates a conference that does not support WebEx. (This is the default value.) *SIP* makes the TelePresence Conductor reserve resources for one call on the primary conference bridge. *SIP-TSP* makes the TelePresence Conductor reserve resources for two calls on the primary conference bridge.
factoryResourceLimits	resourcelimits struct	The resource limits for this conference, made up of **signalling**, **media** and **licences** for a conference bridge of type *tsmcu* and made up of **ports** for a conference bridge of type *mcu*. This parameter is used for Assured scheduling [p.66] and helps to prevent arbitrary conference enlargement.
→ **signalling**	integer	Limit on the number of participants/calls that this conference may use. This is only applicable to conference bridge of type *tsmcu*.
→ **media**	integer	Limit on the media resources that this conference may use. This is only applicable to conference bridge of type *tsmcu*.
→ **licences**	integer	Limit on the licence resources that this conference may use. This is only applicable to conference bridge of type *tsmcu*.
→ **ports**	integer	Limit on number of ports that this conference may use. This is only applicable to conference bridge of type *mcu*.
encryption	string	The encryption requirements for endpoints in this conference. *optional*: endpoints in this conference may have encryption enabled or disabled. (This is the default value.) *required*: endpoints in this conference must have encryption enabled. *forbidden*: endpoints in this conference must not have encryption enabled. This is only applicable to TelePresence Servers, not to TelePresence MCUs.

JSON examples

The following are example **factoryResourceLimits** structs:

- for a TelePresence MCU:

```
factoryResourceLimits:
{
```

```
        "ports": 11
}
```

- for a TelePresence Server:

```
factoryResourceLimits:
{
        "signalling"  : 4 ,
        "media"       : 12288 ,
        "licenses"    : 15360
}
```

Returned data

Parameters	Type	Description
status	string	On success: "operation successful".
factory_conference_id	string	Conference UUID supplied by the TelePresence Conductor - for tracking messages

factory.health.query

This call returns system status information.

Note: clients should avoid calling factory.health.query more frequently than once every 5 minutes. Because of performance implications TelePresence Conductor implements a caching layer, which will not be updated more frequently than once every 5 minutes.

Returned data

Parameters	Type	Description
fan_1	string	The current speed of fan 1 (in RPM and updated approximately once per 5 minutes). Returns N/A if using a VM TelePresence Conductor.
fan_2	string	The current speed of fan 2 (in RPM and updated approximately once per 5 minutes). Returns N/A if using a VM TelePresence Conductor.
fan_3	string	The current speed of fan 3 (in RPM and updated approximately once per 5 minutes). Returns N/A if using a VM TelePresence Conductor.
committed_as	string	The amount of memory that would be used if all the memory that has been allocated were to be used. The committed memory is a sum of all of the memory which has been allocated by processes, even if it has not been used by them as of yet. This is useful if one needs to guarantee that processes will not fail due to lack of memory once that memory has been successfully allocated.
cpu_load_1	integer	The average CPU load taken over a 1 minute period.
cpu_load_5	integer	The average CPU load taken over a 5 minute period.
cpu_load_15	integer	The average CPU load taken over a 15 minute period.

Note that the values reported for **cpu_load_1**, **cpu_load_5** and **cpu_load_15** are Linux CPU loads, as reported in the Linux "uptime" command.

TelePresence Conductor has the following amounts of RAM (to compare to the value of **committed_as**):

- on VM systems (systems that report N/A in the **fan_1** to **fan_3** values) RAM is 6G
- on appliance systems (systems that report anything other than N/A in the **fan_1** to **fan_3** values) RAM is 4G

factory.webex.add

This call adds a WebEx conference to the TelePresence Conductor.

Input parameters

Required parameters

Parameters	Type	Description
conferenceName	string	The name of the conference.
sipUri	string	The SIP URI for the conference. When using the SIP method to create a WebEx conference, the SIP URI is used for both audio and video traffic. When using the SIP-TSP method, the SIP URI is only used for the video traffic. In this case the **audioDN** must be specified too.

Optional parameters

Parameters	Type	Description
bandwidth	integer	The bandwidth (in Kbps) that is required for this conference. The default is 2 MB.
dtmf	string	A string of characters that will be converted to DTMF signals, allowing the device to navigate through audio menus. The sequence may contain 0- 9 , * , # , and , . The comma becomes a two second pause.
audioDn	string	The dial number or URI that is used for audio only when using the SIP-TSP method to create a WebEx conference.

Returned data

None

feedbackReceiver

The TelePresence Conductor API implements the following feedbackReceiver calls:

Note: These calls are deprecated and we do not recommend that you use them. Use the feedbackReceiver.extended [p.14] calls instead.

- **feedback.configure**
- **feedbackReceiver.query**

feedbackReceiver.extended

The API allows you to register an application as a feedback receiver. This means that the application does not have to constantly poll the TelePresence Conductor if it wants to monitor activity.

The device publishes events when they occur, it will send XML-RPC messages to your application's interface when the events occur.

There is a limit of 20 feedback receivers in total.

The call **`feedbackReceiver.extended`** allows clients to register for extended feedback information for specific event notification with pertinent information. This API closely mirrors the existing TelePresence MCU/TelePresence Server feedback APIs. The notification is via the XML-RPC call **`eventNotification`**.

Example notification message:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
  <methodName>eventNotification</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>sourceIdentifier</name>
              <value>
                <string>id1</string>
              </value>
          </member>
          <member>
            <name>seqn</name>
              <value>
                <int>0</int>
              </value>
          </member>
          <member>
            <name>events</name>
            <value>
              <array>
                <data>
                  <value>
                    <struct>
                      <member>
                        <name>name</name>
                        <value>
                          <string>conferenceCreate</string>
                        </value>
                      </member>
                      <member>
                        <name>args</name>
                        <value>
                          <struct>
                            <member>
                              <name>factory_conference_id</name>
                              <value>
                                <string>a7062510-5625-11e1-bdb0-0010f31a4434</string>
```

```
                    </value>
                  </member>
                  <member>
                    <name>request_id</name>
                    <value>
                      <string>a7049626-5625-11e1-ad66-0010f31a4434</string>
                    </value>
                  </member>
                </struct>
              </value>
            </member>
          </struct>
        </value>
      </data>
    </array>
  </value>
 </member>
</struct>
</value>
</param>
</params>
</methodCall>
```

Feedback Events

Feedback events may be sent either from TelePresence Conductor or directly from the underlying conference bridge to the client.

Below is a list of the supported feedback events. Multiple events can be bundled into one XML-RPC notification message and are returned in an array of parameters in the associated **`args`** struct. The parameters documented below can be returned optionally.

Name	Parameters
`conferenceCreate`	factory_conference_id, request_id, conference_name
`conferenceDestroyed`	factory_conference_id
`conferenceJoined`	factory_conference_id, unauthenticated_source_alias, participant_role, request_id Note that the data type for the parameter unauthenticated_source_alias has changed from a string to an array.
`joinCreateRequestReceived`	request_id
`cascadeCreated`	factory_conference_id, request_id
`cascadeDestroyed`	factory_conference_id
`partitionRecovery`	
`joinCreateRequestFailed`	request_id, factory_conference_id
`applicationStarted`	
`alarmsChanged`	

feedbackReceiver.extended.configure

This call configures a feedback receiver on the TelePresence Conductor, which will report particular events that occur on the TelePresence Conductor to the system that is making the request.

Input parameters

Required parameters

Parameters	Type	Description
receiverURI	string	The fully-qualified URI identifying protocol and remote device, for example **http://10.2.134.40:5050/RPC2** Valid protocol types are currently “http” and “https”. If no port number override is specified, then 80 or 443 (respectively) will be used. If a **receiverIndex** has been specified and this parameter is absent or set to the empty string, then the feedback receiver at that index will effectively be de-configured and receive no further notifications.
sourceIdentifier	string	Will be returned in feedback messages (event notifications and **feedbackReceiver.extended.query** messages)

Optional parameters

Parameters	Type	Description
receiverIndex	integer	Which “slot” this receiver should use: If absent assumed to be 1. If <0, then find any available slot (preferred). ReceiverIndex must be in the range 1 – 20 (inclusive)
subscribedEvents	array	An array of strings of event names to subscribe to, where the strings are the names of the notification events. If this parameter is absent, then the receiver will be set up to receive all notifications.

Returned data

Parameters	Type	Description
receiverIndex	integer	Which “slot” has been configured.

Note that if a client tries to configure a feedback receiver using a URI of an existing feedback receiver the call will use the **receiverIndex** of the existing feedbackReceiver.

feedbackReceiver.extended.query

This call returns information about all the feedback receivers that have been configured on the particular TelePresence Conductor.

Input parameters

Required parameters

None

Optional parameters

None

Returned data

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

Parameters	Type	Description
`receivers`	array	The array of receivers containing the following:
→ **`receiverURI`**	string	Fully-qualified URI identifying protocol and remote device, for example **`http://10.2.134.40:5050/RPC2`**
→ **`sourceIdentifier`**	string	A string that is returned in feedback notification events to identify the originator of the notification event. This is provided when configuring the feedback receiver.
→ **`index`**	integer	The index number of the receiver

feedbackReceiver.extended.remove

This call removes the specified feedback receiver from the TelePresence Conductor.

Input parameters

Required parameters

Parameters	Type	Description
`receiverIndex`	integer	The index returned by a **`feedbackReceiver.extended.configure`** request

Optional parameters

None

Returned data

None

Participant

For methods that perform functions on participants in a conference, the unique identifier is **`participantName`**. Where multiple participants in a conference have the same **`participantName`**, the functions are performed on all the participants with that **`participantName`**.

- **participant.disconnect**, for example, will disconnect all participants with the **participantName** specified on the selected conference.
- **participant.modify** will modify all participants with the **participantName** specified on the selected conference.
- **participant.message** will send the message to all participants with the **participantName** specified on the selected conference.
- **participant.diagnostics** and **participant.enumerate** will return the result of doing the function on one of those participants with the same name, the selection of which is completely arbitrary.

participant.add

This call adds a participant to a conference. The participant.add request is forwarded to the appropriate conference bridge on which the conference is hosted. The TelePresence Conductor always enforces the input parameter **participantType** to be *ad_hoc*.

Input parameters

Required parameters

Parameters	Type	Description
conferenceName	string	The name of the conference to add the participant to.
participantName	string	The unique name of the participant to add.
address	string	The address of the endpoint; may be hostname, IP address, E.164 number, SIP URI, or H.323 ID.

Optional parameters

Parameters	Type	Description
factoryNumScreens	string	The number of screens to use on a participant's endpoint. This parameter is only relevant if the conference template associated with the conference that this participant has dialed into: ■ points to a Service Preference containing TelePresence Server pools ■ has **Allow multiscreen** set to *Yes* ■ has a **Maximum screens** value that is greater than the value specified in **factoryNumScreens**. If the conditions above are all met, the number of screens specified on the conference template is overridden with the number specified in **factoryNumScreens**. In all other cases the **factoryNumScreens** parameter is ignored and the number of screens to use on the participant is either set to the **Maximum screens** value on the conference template or set to '1'.
addAsGuest	boolean	Whether the participant is added as a guest on the conference. If the conference is of type Meeting, the parameter defaults to *False* and is rejected when changed to *True*. If the conference is of type Lecture, the parameter defaults to *True* and must be changed to *False* if the participant should be added as a chairperson.

Parameters	Type	Description
participantProtocol	string	The protocol to use for outdial calls: either *sip* or *h323*. In a deployment using the TelePresence Conductor's B2BUA this setting must be *sip*.
participantType	string	The TelePresence Conductor ignores any values specified for this parameter and always enforces the value to be *ad_hoc*.

For conferences hosted on a TelePresence MCU all other optional parameters are documented in *Cisco TelePresence MCU API reference guide*.

Returned data

None

participant.diagnostics

This call returns diagnostic information about a given participant.

Input parameters

Required parameters

Parameters	Type	Description
conferenceName	string	The name of the conference.
participantName	string	The unique name of a participant.
participantProtocol	string	The protocol: *h323* , *sip* or *vnc*.
participantType	string	The type of the participant: *by_address*, *by_name*, or *ad_hoc*.

Optional parameters

None

Returned data

For conferences hosted on a TelePresence MCU the returned data is documented in *Cisco TelePresence MCU API reference guide*. For conferences hosted on a TelePresence Server, the TelePresence Conductor will endeavor to provide similar information.

participant.disconnect

This call causes the connection to the specified participant to be torn down, if such a connection exists.

Note: in TelePresence Conductor version XC2.3 this call was deprecated and replaced by participant.remove [p.24].

Input parameters

Required parameters

Parameters	Type	Description
`conferenceName`	string	The name of the conference.
`participantName`	string	The unique name of a participant.
`participantProtocol`	string	The protocol: *h323* , *sip* or *vnc*.
`participantType`	string	The type of the participant: *by_address*, *by_name*, or *ad_hoc*.

Optional parameters

None

Returned data

For conferences hosted on a TelePresence MCU the returned data is documented in *Cisco TelePresence MCU API reference guide*. For conferences hosted on a TelePresence Server, the TelePresence Conductor will endeavor to provide similar information.

participant.enumerate

This call returns data about all participants in active conferences.

The call works by amalgamating responses for `participant.enumerate` calls to all configured conference bridges.

Input parameters

Required parameters

None

Optional parameters

Parameters	Type	Description
`factoryConferenceIds`	array	An array of `factoryConferenceId` strings. If present, only participants belonging to these conferences will be returned. If absent, details for all participants of all conferences will be returned.

Returned data

Parameters	Type	Description
`participants`	array	An array of participant structs as returned by the conference bridge API.
→ `mcuIPAddress`	string	IP address of the conference device holding the participant. Added by TelePresence Conductor.

Parameters	Type	Description
→ **factoryConferenceId**	string	Internal conference id used to identify which conference the participant belongs to.
→ **participantName**	string	The name of the participant, which does not have to be unique.
→ **participantType**	string	One of *by_address*, *by_name* or *ad_hoc*.
→ **conferenceName**	string	The name of the conference.
→ **address**	string	The address of the endpoint; may be hostname, IP address, E.164 number, SIP URI or H.323 ID.
→ **displayName**	string	The display name of the participant.
→ **factoryBridgeType**	string	One of *tsmcu* or *mcu*.
→ **audioRxMuted**	boolean	*true* means that audio from this participant will not be heard by other conference participants.
→ **audioTxMuted**	boolean	*true* means that the conference bridge does not send the audio part of the conference to this participant.
→ **videoTxMuted**	boolean	*true* means that the conference bridge does not send the video part of the conference to this participant.
→ **dtmfSequence**	string	A string of characters that will be converted to DTMF signals, allowing the device to navigate through audio menus. The sequence may contain 0-9, *, #, and ,. The comma becomes a two second pause.
→ **previewURL**	string	This is only supported by the TelePresence MCU. For the TelePresence Server a blank string is returned.
→ **factoryWebEx**	string	*None* represents a conference that does not support WebEx. (This is the default parameter.) *SIP* represents a conference that supports a SIP WebEx conference, where resources for one call is used on the primary conference bridge. *SIP-TSP* represents a conference that supports a SIP-TSP WebEx conference, where resources for two calls (one for video, one for audio) are used on the primary conference bridge.
→ **factoryWebExCallType**	string	One of *VideoAudio*, *Video* or *Audio*.

Parameters	Type	Description
Additional parameters contained in a **participant** struct, as documented in *Cisco TelePresence MCU API reference guide*.		

Note that the return information about all participants in all conferences (participant information structures for approximately 2,400 participants) might be a bit unwieldy.

participant.message

This call is used to send a message to a participant in their video stream.

Input parameters

Required parameters

Parameters	Type	Description
conferenceName	string	The name of the conference.
participantName	string	The unique name of a participant.
message	string	The string of up to 255 characters to send to the participant.

Optional parameters

Parameters	Type	Description
participantProtocol	string	The protocol: *h323* , *sip* or *vnc*.
participantType	string	The type of the participant: *by_address*, *by_name*, or *ad_hoc*.

Returned data

Parameters	Type	Description
status	string	On success: "operation successful".

participant.modify

This call modifies the active state of a participant in a conference.

Input parameters

Required parameters

Parameters	Type	Description
conferenceName	string	The name of the conference.
participantName	string	The unique name of a participant.
participantProtocol	string	The protocol: *h323* , *sip* or *vnc*.
participantType	string	The type of the participant: *by_address*, *by_name*, or *ad_hoc*.

Optional parameters

Parameters	Type	Description
displayNameOverrideStatus	boolean	*true* if the endpoint uses the **displayNameOverrideValue** text to identify itself to other participants.
displayNameOverrideValue	string	This value overrides the participant's display name if **displayNameOverrideStatus** is *true*.
cpLayout	string	This sets the initial conference view layout for the video sent to the participant.
layoutControlEnabled	boolean	Defines whether the endpoint's participant will have control over the layout.
audioRxMuted	boolean	*true* means that audio from this participant will not be heard by other conference participants.
audioRxGainMode	string	*none*, *automatic*, or *fixed*
audioRxGainMilliDb	integer	If audio gain mode is fixed, this is the number of decibels of gain applied, multiplied by 1000, and can be a negative value.
videoRxMuted	boolean	*true* means that video from this participant will not be seen by other conference participants.
videoTxWidescreen	boolean	If *true*, the TelePresence MCU sends video in a form suitable for a widescreen 16:9 display to this participant.
autoDisconnect	boolean	*true* allows the device to automatically disconnect the endpoint, and all remaining endpoints that have this property, when none of the remaining endpoints require manual disconnection. *false* means this endpoint requires manual disconnection.
suppressDtmf	string	Controls the muting of DTMF tones. One of *fecc*, *always*, *never*, or *default*.
dtmfSequence	string	A string of characters that will be converted to DTMF signals, allowing the device to navigate through audio menus. The sequence may contain 0-9, *, #, and ,. The comma becomes a two second pause.
audioTxMuted	string	*true* means that the conference bridge does not send the audio part of the conference to this participant.
borderWidth	integer	Controls the width of the outer border of a participant's layout. *0* is no border.
important	boolean	*true* means this participant's video is important; it will dominate the layout.

Note that for WebEx conferences only the following parameters are supported:

- **displayNameOverrideStatus**
- **displayNameOverrideValue**
- **videoRxMuted**
- **audioRxMuted**
- **audioTxMuted**

Returned data

For conferences hosted on a TelePresence MCU the returned data is documented in *Cisco TelePresence MCU API reference guide*. For conferences hosted on a TelePresence Server, the TelePresence Conductor will endeavor to provide similar information.

participant.remove

This call causes the connection to the specified participant to be torn down, if such a connection exists.

The call is supported in TelePresence Conductor version XC2.3 or later. It replaces the call participant.disconnect [p.19], which was deprecated in version XC2.3.

Input parameters

Required parameters

Parameters	Type	Description
`conferenceName`	string	The name of the conference.
`participantName`	string	The unique name of a participant.
`participantProtocol`	string	The protocol: *h323* , *sip* or *vnc*.
`participantType`	string	The type of the participant: *by_address*, *by_name*, or *ad_hoc*.

Optional parameters

None

Returned data

For conferences hosted on a TelePresence MCU the returned data is documented in *Cisco TelePresence MCU API reference guide*. For conferences hosted on a TelePresence Server, the TelePresence Conductor will endeavor to provide similar information.

Fault codes

The TelePresence Conductor 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 Conductor and their most common interpretations.

Fault Code	Description
1	`method not supported`. This method is not supported on this device.
2	`duplicate conference name`. A conference name was specified, but is already in use.
3	`duplicate participant name`. A participant 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.

Fault Code	Description
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.
7	**too many participants**. There are already too many participants configured and no more can be created.
8	**no conference name or auto attendant id supplied**. A conference name or auto attendant identifier was required, but was not present.
9	**no participant name supplied**. A participant name is required but was not present.
10	**no participant address supplied**. A participant address is required but was not present.
11	**invalid start time specified**. A conference start time is not valid.
12	**invalid end time specified**. A conference end time is not valid.
13	**invalid PIN specified**. A PIN specified is not a valid series of digits.
14	**authorization failed**. The requested operation is not permitted on this device.
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**. This is in the case that reservedAudioPorts or reservedVideoPorts value is set too high, and the device cannot support this.
18	**duplicate numeric ID**. A numeric ID was given, but this ID is already in use.
19	**unsupported protocol**. A protocol was used which does not correspond to any valid protocol for this method. In particular, this is used for participant identification where an invalid protocol is specified.
20	**unsupported participant type**. A participant type was used which does not correspond to any participant type known to the device.
21	**no conference alias supplied**. A conference alias is required but was not present.
22	**conference.modify 'locked' param unsupported**. The conference.modify parameter 'locked' is not supported in the TelePresence Conductor API.
25	**new port limit lower than currently active**.
26	**floor control not enabled for this conference**.
27	**no such template**. The specified template wasn't found.
30	**unsupported bit rate**. A call tried to set a bit rate that the device does not support.
31	**template name in use**. This occurs when trying to create or rename a template to have the same name as an existing template.
32	**too many templates**. This occurs when trying to create a new template after the limit of 100 templates has been reached.
36	**required value missing**. The call has omitted a value that the TelePresence MCU requires to make the change requested by the call.
42	**port conflict**. The call attempts to set a port number that is already in use by another service.

Fault Code	Description
43	**route already exists**. The call attempts to add a route that has the same **destination** and **prefixLength** as a route that already exists on the TelePresence Conductor.
44	**route rejected**. The call attempts to add a route to a forbidden subnet
45	**too many routes**. The call can not add the route because doing so would exceed the allowed number of routes.
46	**no such route**. The TelePresence Conductor has no record of a route that has the provided **routeId**.
48	**IP address overflows prefix length**. The call attempts to make a route **destination** more specific than the range defined by the **prefixLength**.
49	**operation would disable active interface**.
101	**missing parameter**. This is given when a required parameter is absent. The parameter in question is given in the fault string in the format "missing parameter - parameter_name".
102	**invalid parameter**. This is given when a parameter was successfully parsed, is of the correct type, but falls outside the valid values; for example an integer is too high or a string value for a protocol contains an invalid protocol. The parameter in question is given in the fault string in the format "invalid parameter - parameter_name".
103	**malformed parameter**. This is given when a parameter of the correct name is present, but cannot be read for some reason; for example the parameter is supposed to be an integer, but is given as a string. The parameter in question is given in the fault string in the format "malformed parameter - parameter_ name".
104	**mismatched parameters**. The call provides related parameters that, when considered together, are not expected/supported.
201	**operation failed**. This is a generic fault for when an operation does not succeed as required.

REST APIs

TelePresence Conductor has a number of REST (Representational State Transfer) APIs defined.

The following REST APIs are used by a client such as Cisco TMS to retrieve information about the TelePresence Conductor's configuration and state, and to provision conferences on the TelePresence Conductor:

- Capacity Management API [p.27]
- Provisioning API [p.33]
- SIP Domain API [p.48]

All communication for interacting with the REST APIs is REST/JSON. The 'Content-Type' header should specify 'application/json'.

The REST APIs define the sets of optional and mandatory JSON parameters. If any unknown JSON parameters are received, the API will reject the operation with a message like:

```
"error_code":"20","message":"There was a validation error in the request
which prevented further processing."
```

The error message will indicate the unrecognized fields.

The error codes are defined in Error codes [p.49].

Capacity Management API

The Capacity Management API allows a client to retrieve information on the potential cost of a conference, including the capacity of the conference bridge pool that will be used for the conference. The client sends the conference alias that will be dialed for a conference and receives back the cost and capacity information for the resulting conference.

The main purpose of the Capacity Management API is to provide the API client with the capacity information for a conference to be able to create scheduled conferences and to be able to offer assured scheduling to its users.

The Capacity Management API provides a GET request mechanism where the client can ask for the cost and capacity of a specific conference alias (dial string), for example:

```
GET https://<IP address of the
Conductor>/api/2.3/servicecost/meet.alice@domain.com
```

The response for this request is a JSON data structure that provides the cost and capacity information for the relevant conference. It includes the cost of initializing the conference and the costs for systems with different numbers of screens dialing in.

Access to the Capacity Management API requires user authentication via an admin user with API access privileges configured on the TelePresence Conductor.

Data structures

The Capacity Management API has the following data structures defined:

- ServiceParams object [p.31]
- ParticipantCost object [p.29]

- Resources object [p.28]

The **ServiceParams** data structure is returned to the client when performing a GET request for a particular conference alias. It contains the attributes, capabilities and costs of a conference that would be created if this alias dialed into it. The **ServiceParams** data structure includes a **Resources** data structure defining the capacity of the primary pool in the Service Preference for this conference. The **ServiceParams** data structure also includes the costs of a conference defined by:

- an array of **ParticipantCost** structures to work out the costs for a participant dialing into the conference
- a **Resources** data structure for the required one-off per-conference costs

The **ParticipantCost** data structure is an array of objects describing the costs for a participant depending on the number of screens supported. It contains 1 array element for conference bridges that do not support multiscreen systems and 4 array elements for conference bridges that support multiscreen systems. The **ParticipantCost** data structure includes a **Resources** data structure for each possible number of endpoint screens supported.

The **Resources** data structure is an array of name-value-pair objects describing the resource costs associated with the dimensions of resource calculations applicable to the conference bridge type.

Resources object

The **Resources** data structure is an array of name-value-pair objects describing the resource costs associated with the dimensions of resource calculations applicable to the conference bridge type. The **Resources** object can be used to describe the overall costs associated with hosting a conference on a particular conference bridge type based on how the TelePresence Conductor is configured.

Considerations for determining resources

The Capacity Management API provides the number of resources that are potentially available from conference bridges and conference bridge pools in an indicated Service Preference, based on how the TelePresence Conductor is currently configured. The API does not provide information about the resource usage for conferences that are currently running. Changes to the TelePresence Conductor configuration may result in different resource information.

The dimensions for determining the resource usage differ depending on the conference bridge type that is used:

- For TelePresence MCUs there is only one dimension, namely *ports*.
- For TelePresence Servers there are three dimensions:
  - signalling
  - media
  - licences

Conference bridges that have their **Status** set to *Unusable* are included in the resource calculation, since they are assumed to be only temporarily out of service.

Attributes

Attribute name	Type	Description
dimension	string	The dimension of the resources presented. For a TelePresence MCU, the dimension must be *ports*. For a TelePresence Server the dimension can be one of *signalling*, *media* or *licences*. The string must be less than 128 characters long.
value	integer	The cost of this resource.

JSON example

For a TelePresence MCU the JSON array could be:

```
[
        { "dimension": "ports", "value": 80 }
]
```

For a TelePresence Server the JSON array could be:

```
[
        { "dimension": "signalling", "value": 100 },
        { "dimension": "media", "value": 200 },
        { "dimension": "licenses", "value": 300 }
]
```

ParticipantCost object

The **ParticipantCost** data structure is an array of objects describing the costs for a participant depending on the number of screens supported.

For conferences hosted on conference bridges that do not support multiscreen endpoints, the array only contains a single element. The **num_screens** value of this array element is 1.

For conferences hosted on conference bridges that support multiscreen endpoints, the array contains 4 elements, each with a different **num_screens** value.

Attributes

Attribute name	Type	Description
num_screens	integer	The number of screens that a participant could have and to which the costs in the **Resources** object apply. For conferences hosted on conference bridges that do not support multiscreen endpoints, the value is always 1. For conferences hosted on conference bridges that support multiscreen endpoints, the value will be in the range 1 to 4. Each of the 4 array elements will have a different **num_screens** value.
resources	**Resources** object	The costs that would apply if the participant had the number of screens in **num_screens**.

JSON example

For a conference hosted on a TelePresence MCU the JSON array could be:

```
[
        {
                "num_screens": 1,
                "resources": [
                        { "dimension": "ports", "value": 80 }
                ]
        },
]
```

For a conference hosted on a TelePresence Server the JSON array could be:

```
[
        {
                "num_screens": 1,
                "resources": [
                        { "dimension": "signalling", "value": 150 },
                        { "dimension": "media",      "value": 250 },
                        { "dimension": "licenses",   "value": 350 }
                ]
        },
        {
                "num_screens": 2,
                "resources" : [
                        { "dimension": "signalling", "value": 250 },
                        { "dimension": "media",      "value": 450 },
                        { "dimension": "licenses",   "value": 650 }
                ]
        },
        {
                "num_screens": 3,
                "resources" : [
                        { "dimension": "signalling", "value": 350 },
                        { "dimension": "media",      "value": 650 },
                        { "dimension": "licenses",   "value": 950 }
                ]
        },
```

```
        {
                "num_screens": 4,
                "resources" : [
                        { "dimension": "signalling", "value": 450 },
                        { "dimension": "media",      "value": 850 },
                        { "dimension": "licenses",   "value": 1250 }
                ]
        }
]
```

ServiceParams object

The **`ServiceParams`** data structure is returned to the client when performing a GET request for a particular conference alias. It contains the attributes, capabilities and costs of a conference that would be created if this alias dialed into it. The information that is returned in the **`ServiceParams`** data structure is based on the current configuration of the TelePresence Conductor. It does not take into account the resource usage for conferences that are currently running. Changes to the TelePresence Conductor configuration may result in different resource information.

All strings must be less than 1024 characters unless specified otherwise.

Attributes

Attribute name	Type	Description
`id`	string	The UUID (a globally unique ID) of the Service Preference that would be used for the given alias.
`display_name`	string	The display name defined for the Service Preference.
`description`	string	The description defined for the Service Preference.
`alias`	string	The alias string received from the client.
`bridge_type`	string	The enum string for the type of conference bridge associated with this Service Preference. This is *mcu* for a TelePresence MCU or *tsmcu* for a TelePresence Server.
`bridge_ capabilities`	array [string]	An array of capabilities that are supported by the conference bridge. The strings *cascading* and *multiscreen* are included in the array if the capabilities are supported by the conference bridge. The strings are excluded from the array if the capabilities are not supported.

Attribute name	Type	Description
max_screens	integer	The value for **Maximum screens** that has been defined on the associated conference template. This can be in the range of 1 to 4. For TIP-compliant endpoints dialing into a rendezvous conference using the TelePresence Conductor's B2BUA the value is the maximum limit of supported screens. For reserved chairperson participants, policy service routed calls and ad hoc escalated calls the value is the default number of screens for initial resource calculations. The value is always 1 for conference bridges that do not support multiscreen endpoints (where **bridge_capabilities** does not include *multiscreen*).
max_participants_per_conference_min	integer	The lowest max_participants_per_conference that is supported for a conference hosted through this Service Preference. The value must be 1 or more.
max_participants_per_conference_max	integer	The highest max_participants_per_conference that is supported for a conference hosted through this Service Preference. The value must be 1 or more.
max_participants_per_conference_template	integer	The value defined for **Maximum** when limiting the number of participants on the associated conference template. 0 means that there is no limit set. The value includes any auto-dialed participants and reserved chairperson participants defined for the conference template.
pool_capacity	array [Resources]	A JSON object containing the resource capacity, which is potentially available in the primary conference bridge pool associated with the Service Preference that would be used by the specified alias. Fallback pools are not considered in this computation.
conference_cost	array [Resources]	A JSON object indicating any one-off per-conference costs.
participant_cost	array [ParticipantCost]	An array of JSON objects containing the total costs for the participant taking into consideration multiple screens. If **bridge_capabilities** includes *multiscreen*, there will be 4 elements in the array. If **bridge_capabilities** does not include *multiscreen*, there will only be 1 element in the array. The client can use the appropriate array element for the actual number of screens to work out the required costs.

ServiceCost operation

ServiceCost is a REST operation. It only includes a GET method to retrieve a **ServiceParams** data structure for a given alias. It does not include any PUT, POST or DELETE methods.

The request URI is **/api/2.3/servicecost/<alias>**

The alias in the request URI represents a dial string. It must not include a regular expression and it must not be more than 1024 characters long.

Sample JSON

A valid GET operation for the alias string **`meet.alice@cisco.com`** returns a JSON string like this:

```
{
        "id": "58e26522-da56-11e2-b4f8-0010f31595fc",
        "display_name": "SP_1_North_America",
        "description": "Prefer MCUs in the USA.",
        "alias": "meet.alice@cisco.com",
        "bridge_type": "mcu",
        "bridge_capabilities": ["cascading"],
        "max_screens": 1,
        "max_participants_per_conference_min": 7,
        "max_participants_per_conference_max": 18,
        "max_participants_per_conference_template": 4,
        "pool_capacity": [
                { "dimension": "ports", "value": 80 }
        ],
        "conference_cost": [
                { "dimension": "ports", "value": 2 }
        ],
        "participant_cost" : [
                {
                        "num_screens": 1,
                        "resources": [
                                { "dimension": "ports", "value": 1 }
                        ]
                }
        ]
}
```

Provisioning API

The Provisioning API allows a client such as Cisco TMS to provision collaboration meeting rooms (CMRs) on the TelePresence Conductor. The API is a REST API and uses JSON for all communication.

The main purpose of the Provisioning API is for the client to perform a PUT ConfBundle operation. A ConfBundle contains the data required to create a conference for one or more end-users, including conference template information, a set of conference aliases, a set of auto-dialed participants and a conference name. Because some of the information that must be included in the ConfBundle data structure is unknown to the client, the client must first perform GET operations for QualityInfo and ServiceInfo. These

data structures contain information about the quality settings and the Service Preference settings, respectively, that are available on this TelePresence Conductor.

Data structures

The Provisioning API has the following data structures defined:

- QualityInfo object [p.34]
- ServiceInfo object [p.36]
- ConfBundleId object [p.37]
- ConfBundleVer object [p.37]
- ConfBundle object [p.38]
- Alias object [p.44]
- AutoDialedParticipant object [p.45]

The **`QualityInfo`** data structure is a JSON array of **`QualitySpec`** objects, containing all quality settings that have been defined on this TelePresence Conductor.

The **`ServiceInfo`** data structure is a JSON array of **`ServiceRecord`** objects, containing all Service Preferences defined on this TelePresence Conductor. The **`ServiceRecord`** object includes conference bridge information and a reference to a group of qualities contained in the **`QualityInfo`** object.

Not all quality settings are applicable to all conference bridge types or Service Preferences. To work out the subset of quality settings that are applicable to a specific **`ServiceInfo`**, query the **`QualityInfo`** object specifying **`ServiceInfo.qualities`** as the quality group in the element URI.

The **`Alias`** data structure is a JSON object describing a dial string that will cause a participant to join a particular conference in a particular role.

The **`AutoDialedParticipant`** data structure is a JSON object describing a participant that the system will dial when a particular conference is created.

The **`ConfBundle`** data structure contains information required by TelePresence Conductor to be able to provision a conference. It includes conference template information, a set of **`Alias`** objects, a set of **`AutoDialedParticipant`** objects and a conference name. The ConfBundle data structure can be saved to the TelePresence Conductor with a PUT operation or it can be retrieved from the TelePresence Conductor with a GET operation.

`ConBundleId` and **`ConfBundleVer`** uniquely identify a specific revision of a stored **`ConfBundle`**.

QualityInfo object

The **`QualityInfo`** data structure is a JSON array of **`QualitySpec`** objects, containing all quality settings that have been defined on this TelePresence Conductor.

All strings must be less than 1024 characters unless specified otherwise.

Attributes

The **`QualityInfo`** data structure contains the following attributes:

Attribute name	Type	Description
`quality_group_name`	string	An enum string that identified this set of quality settings, for example **`qualgroup_ts`**.

Attribute name	Type	Description
quality_group	array [QualitySpec]	An array of **QualitySpec** objects as defined below.

The **QualitySpec** object contains the following attributes:

Attribute name	Type	Description
quality_id	string	The UUID (a globally unique ID) of the quality setting. This ID is used within the **content_quality**, **host_quality** and **guest_quality** fields of the **ConfBundle** data structure.
display_name	string	A descriptive name for this quality setting, for example **Full HD (1080p 30fps / 720p 60fps video, multi-channel audio)**.
quality_type	string	An enum string that identifies the type of quality setting, either *content* or *video*.
display_order	integer	This number can be used by the client to compare or order **QualitySpec** objects. Larger numbers indicate a lower resource usage. If two qualities have the same **display_order** value, they can be further ranked alphabetically. This number is generated by TelePresence Conductor. The number is greater than 0.

JSON example

The following example **QualityInfo** array contains one **QualitySpec** object for a TelePresence Server and one for a TelePresence MCU:

```
[
        {
        "quality_group_name": "qualgroup_ts",
        "quality_group": [
                {
                "quality_id": "4dd69ffc-f2f7-11e0-8fa5-000c3907bedd",
                "display_name": "Full HD (1080p 30fps / 720p 60fps video, multi-channel audio)",
                "quality_type": "video",
                "display_order": 1
                },
                {
                "quality_id": "53c74afc-a2a7-41f0-7faa-000c4a07bf21",
                "display_name": "HD (720p 30fps video, stereo audio)",
                "quality_type": "video",
                "display_order": 2
                },
                {
                "quality_id": "4ab69faa-f2f7-11e0-8fa5-000c3907bedd",
                "display_name": "1280 x 720p 15fps",
```

```
                "quality_type": "content",
                "display_order": 3
                }
        ]
        },
        {
        "quality_group_name": "qualgroup_mcu",
        "quality_group": [
                {
                "quality_id": "0af69ffc-e3f7-12a0-80a5-000ba977a3fa",
                "display_name": "MCU Quality bundle",
                "quality_type": "video",
                "display_order": 1
                }
        ]
        }
]
```


ServiceInfo object

The **ServiceInfo** data structure is a JSON array of **ServiceRecord** objects, containing all Service Preferences defined on this TelePresence Conductor. The **ServiceRecord** object includes conference bridge information and a reference to a group of qualities contained in the **QualityInfo** object.

All strings must be less than 1024 characters unless specified otherwise.

Attributes

The **ServiceInfo** data structure contains an array of **ServiceRecord** objects.

The **ServiceRecord** object contains the following attributes:

Attribute name	Type	Description
service_id	string	The UUID (a globally unique ID) of the Service Preference. This ID is used within the **service_preference_id** field of the **ConfBundle** data structure.
display_name	string	A descriptive name for this Service Preference.
description	string	A description of this Service Preference.
bridge_type	string	An enum string that identifies the conference bridge type: **tsmcu** for TelePresence Server or **mcu** for TelePresence MCU.
qualities	string	The name that identifies a list of quality settings, for example **qualgroup_ts**. This quality name is used to index into the **QualityInfo** object. Only qualities from this list may be used in conjunction with this **ServiceInfo** data structure.

Attribute name	Type	Description
bridge_capabilities	array [string]	An array of capabilities that are supported by the conference bridge. The strings *cascading* and *multiscreen* are included in the array if the capabilities are supported by the conference bridge. The strings are excluded from the array if the capabilities are not supported.

JSON example

The following example **ServiceInfo** array contains one **ServiceRecord** object for a TelePresence Server and one for a TelePresence MCU:

```
[
        {
                "service_id":"123456",
                "display_name": "SD MCUs",
                "description": "Service Preference for SD MCUs",
                "bridge_type": "mcu",
                "qualities": "qualgroup_mcu",
                "bridge_capabilities": ["cascading"]
        },
        {
                "service_id":"13579",
                "display_name": "HD TSs",
                "description": "Service Preference for HD TelePresence Servers",
                "bridge_type": "tsmcu",
                "qualities": "qualgroup_ts",
                "bridge_capabilities": ["multiscreen"]
        }
]
```

ConfBundleId object

The **ConfBundleID** is used to identify a **ConfBundle** object. It is a string generated by the client when creating a new ConfBundle object. It is returned when a client retrieves the list of all **ConfBundle** objects and it can be provided by the client to retrieve or replace a particular **ConfBundle** object.

It can be any string, as long as it is unique for each ConfBundle. We recommend using a globally unique identifier (GUID) when implementing the API client.

ConfBundleVer object

The **ConfBundleVer** data structure describes the current version of a particular **ConfBundle** object. It is returned when a client retrieves the dictionary of all **ConfBundle** objects.

At a given time, a **ConfBundle** only has a single **ConfBundleVer**; if a **ConfBundle** is updated (with a PUT) it will be given a new **ConfBundleVer** and the old **ConfBundleVer** will cease to have any significance. A **ConfBundleVer** only has significance for a given **ConfBundleId**.

ConfBundle object

The **ConfBundle** data structure contains information required by TelePresence Conductor to be able to provision a conference. It includes conference template information, a set of **Alias** objects, a set of **AutoDialedParticipant** objects and a conference name. The ConfBundle data structure can be saved to the TelePresence Conductor with a PUT operation or it can be retrieved from the TelePresence Conductor with a GET operation.

Provisioned conferences are of type 'lecture', which means that there are two role types defined for **Alias** objects, each with different privileges. We recommend that you configure at least one **Alias** object with a role of *host* and at least one **Alias** object with a role of *guest* for each **ConfBundle** object. The Provisioning API does not enforce this requirement, but it may enforce it in a future version of the TelePresence Conductor.

All strings must be less than 1024 characters unless specified otherwise.

Attributes

The **ConfBundle** data structure contains the following attributes:

Attribute name	Type	Description	Mandatory or optional
conference_display_name	string	The name of the conference to be created. This string need not be unique. The conference display name is displayed on the endpoint screen if the conference is hosted on a TelePresence Server.	mandatory
service_preference_id	string	The UUID (a globally unique ID) pointing to the Service Preference that the TelePresence Conductor should use for this conference. This Service Preference must have been configured on the TelePresence Conductor. The ID is taken from the **service_id** field of the corresponding **ServiceInfo** object.	mandatory
aliases	array [Alias]	An array of **Alias** objects indicating all alias strings that result in this conference. This array contains between 1 and 10 elements.	mandatory
max_participants	integer	The maximum number of participants for the conference, including auto-dialed and reserved participants. The value must be '0' or between 2 and 2400. The value '1' is not accepted. If the value is '0' or omitted, it is implied that there is no limit to the number of participants.	optional
max_duration	integer	The maximum duration of the conference in minutes. The value must be between 0 and 10080. If the value is '0' or omitted, it is implied that there is no limit to conference duration.	optional
allow_content	boolean	Whether the conference allows for content to be shared. If the attribute is omitted, *false* is implied.	optional

Attribute name	Type	Description	Mandatory or optional
content_quality	string	The UUID (a globally unique ID) pointing to the content quality to be used for this conference. The ID is selected from the **quality_id** field within the **QualitySpec** data structure. The quality selected must be from the subset of qualities applicable to the given Service Preference (**QualityInfo [ServiceInfo.qualities]**). The **quality_type** attribute of the **QualitySpec** data structure must have the value *content*. This attribute is ignored if **allow_content** is *false*.	mandatory (if **allow_content** is *true*)
host_quality	string	The UUID (a globally unique ID) pointing to the chairperson quality to be used for this conference. The ID is selected from the **quality_id** field within the **QualitySpec** data structure. The quality selected must be from the subset of qualities applicable to the given Service Preference (**QualityInfo [ServiceInfo.qualities]**). The **quality_type** attribute of the **QualitySpec** data structure must have the value *video*.	mandatory
guest_quality	string	The UUID (a globally unique ID) pointing to the guest quality to be used for this conference. The ID is selected from the **quality_id** field within the **QualitySpec** data structure. The quality selected must be from the subset of qualities applicable to the given Service Preference (**QualityInfo [ServiceInfo.qualities]**). The **quality_type** attribute of the **QualitySpec** data structure must have the value *video*.	mandatory

Attribute name	Type	Description	Mandatory or optional
layout	string	An enum string that defines the video layout scheme to be seen by participants joining conferences created from this **ConfBundle**. The **layout** must be one of the following types, which have been defined in this API: ■ *equal*: conference participants are shown in a grid pattern of equal sized panes, up to 4x4. (Not applicable to multiscreen endpoints) ■ *active*: the active speaker is shown in a large pane with additional participants appearing in up to nine PIPs (picture-in-pictures) overlaid at the bottom of the screen. ■ *prominent*: the active speaker is shown in a large pane with additional participants appearing in up to four smaller panes at the bottom of the screen. (Not applicable to multiscreen endpoints) ■ *single*: the active speaker is shown in one full-screen pane. Depending on the conference bridge capabilities, the closest approximation to the specified layout will be used. Where applicable, multiscreen systems will be mapped to the closest approximation to the specified layout. See **Conference layouts** in the *Cisco TelePresence Conductor Administrator Guide* or Online Help for more information on layout options available on the conference bridge types. This is an optional parameter; if it is not provided, the default conference bridge layout will be used. If the parameter is not provided in a PUT method, a subsequent GET of the **ConfBundle** will return a JSON **null**.	optional
advanced_ parameters	Object	A conference bridge specific JSON object for configuring advanced conference parameters on conference creation. The JSON object must be valid. The TelePresence Conductor does not perform any in-depth checking of data. See Conference bridge specific advanced parameters [p.42] for more information about prohibited and discouraged parameters.	optional
allow_ multiscreen	boolean	Whether or not this **ConfBundle** will allow for multiscreen systems to be displayed in the conference. This attribute is ignored if the **bridge_capabilities** attribute within the **ServiceInfo** data structure does not include *multiscreen*. If **allow_multiscreen** is omitted, the value is implied to be *false*.	optional

Attribute name	Type	Description	Mandatory or optional
max_num_of_screens	integer	The maximum number of screens an endpoint is allowed to have; in the range of 1 to 4. This attribute is ignored if **allow_multiscreen** is *false*. If **max_num_of_screens** is omitted, it is implied to be '1'.	optional
optimize_resources	boolean	Whether or not to allow TelePresence Conductor to optimize the resources used by participants in the conference. This attribute is ignored if the **bridge_type** attribute for the **ServiceInfo** data structure is *mcu*. If **optimize_resources** is omitted, it is implied to be *true*.	optional
reserved_hosts	integer	The number of chairperson participants for whom resources should be reserved. If **reserved_hosts** is omitted, it is implied to be '1'.	optional
reserved_cascades	integer	The number of cascade ports to reserve for this conference. This attribute is ignored if the **bridge_capabilities** attribute within the **ServiceInfo** data structure does not include *cascading*.If **reserved_cascades** is omitted, it is implied to be '1'.	optional
cascade_advanced_parameters	Object	A conference bridge specific JSON object for configuring advanced conference parameters for cascaded conferences. The JSON object must be valid. The TelePresence Conductor does not perform any in-depth checking of data. See Conference bridge specific advanced parameters [p.42] for more information about prohibited and discouraged parameters. The cascade advanced parameters must not include the parameters **pin** and/or **guestPin**. You cannot configure the Pins used on a cascade conference bridge to be different from the ones used on the primary conference bridge. The Pin(s) needed for entry to the cascade bridge will always be the same as the Pin(s) required for the primary bridge. This attribute is ignored if the **bridge_capabilities** attribute within the **ServiceInfo** data structure does not include *cascading*.	optional
scheduled	boolean	Whether or not conferences generated from this **ConfBundle** are scheduled. If the value is *true* the conference can only be created via the API call **factory.conferencecreate** and not by participants dialing the conference alias. If **scheduled** is omitted, it is implied to be *false*.	optional

Attribute name	Type	Description	Mandatory or optional
auto_dialed_participants	array [AutoDialedParticipant]	An array of **AutoDialedParticipant** objects, with 1 to 10 elements. If **auto_dialed_participants** is omitted, it is implied that there are no auto-dialed participants in this conference.	optional
host_pin	string	The PIN required for chairperson participants to join the conference. If **host_pin** is omitted, it is implied that there is no PIN required.	optional
guest_pin	string	The PIN required for guest participants to join the conference. If **guest_pin** is omitted, it is implied that there is no PIN required.	optional
conference_name	string	The unique conference name that TelePresence Conductor generates from the **conference_display_name** provided by the client. The TelePresence Conductor appends a 6 digit number that is random and unique to the **conference_display_name**. On a TelePresence MCU, if the **conference_display_name** is longer than 32 characters, it is truncated at 32 characters and the 6 digit number is appended. The **conference_name** changes every time the confBundle is updated. It is used to create a conference on the conference bridge. The **conference_name** attribute must not be supplied in a PUT operation, but it is included as a read-only attribute in a GET operation.	N/A
version_id	string	The unique conference version ID that TelePresence Conductor generates when the **confBundle** is created or updated. The **version_id** attribute must not be supplied in a PUT operation, but it is included as a read-only attribute in a GET operation.	N/A

Conference bridge specific advanced parameters

The conference bridge specific advanced parameters must be provided as a valid JSON object within the advanced_parameters and (if applicable) cascade_advanced_parameters attributes of the ConfBundle data structure. The sections below list the discouraged and prohibited advanced parameters for each conference bridge type.

TelePresence MCU advanced parameters

When specifying the advanced parameters or cascade advanced parameters for a TelePresence MCU do not use the following parameters. They are rejected by the TelePresence Conductor API because changing them will result in a failure to create conferences.

Prohibited and rejected parameters:

- **conferenceName**
- **numericId**
- **guestNumericId**
- **startTime**
- **maximumAudioPorts**
- **reservedAudioPorts**
- **maximumVideoPorts**
- **reservedVideoPorts**
- **enforceMaximumAudioPorts**
- **enforceMaximumVideoPorts**
- **repetition**
- **weekday**
- **whichWeek**
- **weekDays**
- **terminationType**
- **terminationDate**
- **numberOfRepeats**

We recommend that you also do not specify the following TelePresence MCU parameters, because changing them may result in a failure to create conferences.

Discouraged parameters:

- **cleanupTimeout**
- **contentMode** (do not use when running TelePresence MCU version 4.2)
- **contentContribution**
- **h239Enabled**
- **durationSeconds**
- **private**
- **pin**
- **guestPin**

TelePresence Server advanced parameters

When specifying the advanced parameters for a TelePresence Server do not use the following parameters. They are rejected by the TelePresence Conductor API because changing them will result in a failure to create conferences.

Prohibited and rejected parameters:

- **conferenceName**
- **conferenceReference**
- **startTime**
- **metadata**

We recommend that you also do not specify the following TelePresence Server parameters, because changing them may result in a failure to create conferences.

Discouraged parameters:

- **conferenceMediaTokens**
- **conferenceMediaTokensUnlimited**
- **conferenceMediaCredits**
- **conferenceMediaCreditsUnlimited**
- **waitForChair**
- **duration**
- **durationUnlimited**
- **maxParticipants**
- **maxParticipantsUnlimited**
- **pin**

Alias object

The **Alias** data structure is a JSON object describing a dial string that will cause a participant to join a particular conference in a particular role.

All strings must be less than 1024 characters unless specified otherwise.

We recommend that you configure at least one **Alias** object with a role of *host* and at least one **Alias** object with a role of *guest* for each **ConfBundle** object. The Provisioning API does not enforce this requirement, but it may enforce it in a future version of the TelePresence Conductor.

Attributes

The **Alias** object contains the following attributes:

Attribute name	Type	Description	Mandatory or optional
role	string	An enum string describing the role for this conference alias; either *host* or *guest*.	mandatory
exact_match	string	The exact match alias for the corresponding participant, for example **meet.alice@domain.com**. The supplied dial string must exactly match the alias. The **exact_match** string must be unique on this TelePresence Conductor. This string must be supplied to TelePresence Conductor in lower case or else it will be rejected. The dial string used by participants when dialing into a conference can be in any case; TelePresence Conductor will convert it to a lower case string before matching it to the **exact_match** string stored. Unified CM appends its IP address to dial strings. For a dial string to exactly match the **exact_match** string defined for this alias, the TelePresence Conductor uses the SIP Domain API [p.48] to replace the IP address with the SIP domain.	mandatory

Attribute name	Type	Description	Mandatory or optional
allow_conference_creation	boolean	Whether participants dialing this conference alias can create the conference or not. If the value is *false* the conference must be created via the API call **factory.conferencecreate** or via a second alias that matches to the same conference and has **allow_conference_creation** set to *true*. If **allow_conference_creation** is omitted, it is implied to be *true*.	optional

AutoDialedParticipant object

The **AutoDialedParticipant** data structure is a JSON object describing a participant that the system will dial when a particular conference is created.

All strings must be less than 1024 characters unless specified otherwise.

Attributes

The **AutoDialedParticipant** object contains the following attributes:

Attribute name	Type	Description	Mandatory or optional
role	string	An enum string describing the role for this auto-dialed participant; either *host* or *guest*.	mandatory
participant_quality	string	The UUID (a globally unique ID) pointing to the maximum quality to be used for this auto-dialed participant. The ID is selected from the **quality_id** field within the **QualitySpec** data structure. The quality selected must be from the subset of qualities applicable to the given Service Preference (**QualityInfo [ServiceInfo.qualities]**). The **quality_type** attribute of the **QualitySpec** data structure must have the value *video*.	mandatory
keep_conference_alive	boolean	Whether or not the participant should keep the conference alive. The conference will terminate after the last dial-in participant has left, unless there is an auto-dialed participant with this parameter set to *true*. If **keep_conference_alive** is omitted, it is implied to be *false*.	optional
dtmf	string	The DTMF sequence, which the system will dial when connecting to the conference. The DTMF sequence can include the digits 0-9 and the characters '*' '#' and ',' (comma). It must not be longer than 31 characters.	optional
enabled_state	boolean	Whether or not this auto-dialed participant is enabled. If the value is *disabled*, the auto-dialed participant will not be dialed when the conference is created. If **enabled_state** is omitted, it is implied to be *enabled*.	optional

Note that the **participant_protocol** attribute cannot be changed, it is always set to SIP.

ServiceInfo operation

ServiceInfo is a REST operation. It includes two GET methods to retrieve either an array of **ServiceRecord** objects or an individual **ServiceRecord**. It does not include any PUT, POST or DELETE methods.

Retrieving a list of ServiceRecord objects

The collection URI for retrieving a JSON array of **ServiceRecord** objects is:

/api/2.3/serviceinfo/

Retrieving an individual ServiceRecord object

The element URI for retrieving a specific **ServiceRecord** object with a given ID is:

/api/2.3/serviceinfo/<ServiceRecord.id>

QualityInfo operation

QualityInfo is a REST operation. It includes two GET methods to retrieve either the **QualityInfo** object or an array of **QualitySpec** objects for a given **quality_group_name**. It does not include any PUT, POST or DELETE methods.

Retrieving a QualityInfo object

The collection URI for retrieving a **QualityInfo** object, containing information about all quality settings configured on this TelePresence Conductor, is:

/api/2.3/qualities/

Retrieving an array of QualitySpec objects

The element URI for retrieving a specific **QualitySpec** object, containing the quality settings for a given **quality_group_name**, is:

/api/2.3/qualities/<QualityInfo.quality_group_name>

ConfBundles object

ConfBundles is a REST operation. It includes two GET methods to retrieve either a dictionary of the **ConfBundleID** and **ConfBundleVer** pairs for all **ConfBundle** objects or a specific **ConfBundle** object for a given **ConfBundleID**.

It also includes a PUT method for a specific **ConfBundle** object, and DELETE methods for all **ConfBundle** objects or a specific **ConfBundle** object.

It does not include any POST methods.

Retrieving a dictionary of all ConfBundle objects

To retrieve a dictionary of all **ConfBundle** objects (as a JSON object containing unique **ConfBundleID**s that point to the corresponding **ConfBundleVer**) the client must perform the following operation:

GET **/api/2.3/confbundles/**

This is an example of the JSON that is returned when a client performs this GET operation (without specifying a UUID):

```
{
```

```
        "b4c761e6-a32f-4909-b33f-244862180906": "4a2b393a-910d-435f-946b-31e34813bb9b",
        "4b54f5ff-2bc2-4823-8173-fdd460b51279": "d0e475eb-c657-4849-a436-935633ab2d55",
        "c218e5eb-589a-4d1c-84f6-46e3874250ce": "606297c5-4d60-44c8-962c-1a9a6fba4fb8"
}
```

Retrieving a specific ConfBundle object

To retrieve a specific **ConfBundle** object with its associated attributes the client must perform the following operation:

GET **/api/2.3/confbundles/<ConfBundle.conf_bundle_id>**

where **ConfBundle.conf_bundle_id** is the **ConfBundleID** for a specific **ConfBundle** object.

If the addressed **ConfBundle** does not exist, a **404 Not Found** error will be returned.

Deleting all ConfBundle objects

To delete all **ConfBundle** objects on this TelePresence Conductor the client must perform the following operation:

DELETE **/api/2.3/confbundles/**

Note that this operation will remove all ConfBundle information from the TelePresence Conductor. In the case of large data sets, this will be a highly disruptive operation. Normal operation of the TelePresence Conductor in creating, managing and deleting conferences is likely to be impacted. This operation should only be used with great care.

Deleting a specific ConfBundle object

To delete a specific **ConfBundle** object the client must perform the following operation:

DELETE **/api/2.3/confbundles/<ConfBundle.conf_bundle_id>**

where **ConfBundle.conf_bundle_id** is the **ConfBundleID** for a specific **ConfBundle** object.

If the addressed **ConfBundle** does not exist, a **404 Not Found** error will be returned.

Creating or replacing a specific ConfBundle object

- To create a new ConfBundle object the client must perform the following operation and pass in a ConfBundle object:
  PUT **/api/2.3/confbundles/<ConfBundle.conf_bundle_id>**
  The TelePresence Conductor will create the ConfBundle object with the ConfBundleId set to the value supplied in ConfBundle.conf_bundle_id and return a ConfBundleVer. The API will also return an HTTP response of "201, Created".
- To replace a ConfBundle object the client must perform the following operation and pass in a ConfBundle object with all attributes filled in (those that should change and those that should remain the same):
  PUT **/api/2.3/confbundles/<ConfBundle.conf_bundle_id>**
  where **ConfBundle.conf_bundle_id** is a **ConfBundleId** that is already linked to a **ConfBundle** on the TelePresence Conductor.
  The TelePresence Conductor will replace the **ConfBundle** object and return a new **ConfBundleVer**.
  The API will also return an HTTP response of "200, OK".

SIP Domain API

The SIP Domain API allows a client to set and retrieve the SIP domain that is configured on the TelePresence Conductor.

The main purpose of the SIP Domain API is to work around the following situation:

- Unified CMs append the TelePresence Conductor's IP address (configured as an additional IP address) or hostname instead of the domain to numeric dial strings. For example, when an endpoint dials the string **`1234`**, Unified CM will send the dial string **`1234@10.0.0.1`** to the TelePresence Conductor.
- When TelePresence Conductor attempts to do an exact match of the dial string, it will not be able to match the dial string to an alias, because the user will have provisioned an alias that uses a domain, for example **`1234@domain.com`**.

By setting a domain on TelePresence Conductor and subsequently transforming incoming dial strings to include the domain instead of the TelePresence Conductor's IP address or hostname, it is possible for TelePresence Conductor to exactly match a dial string to a provisioned alias.

SIPDomainSpec object

The **`SIPDomainSpec`** data structure is a JSON object containing the **`SIPDomain`** attribute.

Attributes

Attribute name	Type	Description
`SIPDomain`	string or null	The SIP domain configured on this TelePresence Conductor. If TelePresence Conductor receives a dial string with an IP address appended from Unified CM, the IP address will be replaced with the TelePresence Conductor's SIP domain. The SIPDomain can be either: ■ a JSON string of between 1 and 253 characters representing a SIP domain (only letters, digits, hyphens, underscores and periods are supported) ■ a JSON null value indicating that the SIP domain on this TelePresence Conductor is not set

JSON example

The following example **`SIPDomainSpec`** JSON object contains a SIP domain:

```
{
        "SIPDomain": "100.example-name.com"
}
```

The following example **`SIPDomainSpec`** JSON object contains a null JSON object:

```
{
        "SIPDomain": null
}
```

SIPDomainSpec operation

`SIPDomainSpec` is a REST operation. It includes a PUT method to define the SIPDomain attribute and a GET method to retrieve a **`SIPDomainSpec`** data structure. It does not include any POST or DELETE methods.

Retrieving a SIPDomainSpec object

To retrieve a **`SIPDomainSpec`** object, containing information about the SIP domain configured on this TelePresence Conductor, the client must perform the following operation:

GET **`/api/2.3/sipdomain`**

Replacing the TelePresence Conductor's SIPDomainSpec

To replace the SIPDomainSpec defined on the TelePresence Conductor, the client must perform the following operation:

PUT **`/api/2.3/sipdomain`**

The client must provide a **`SIPDomainSpec`** JSON object.

On success, the PUT method will return a **`204 No Content`** StatusCode. On a request error, a **`400 BadRequest`** StatusCode will be returned, and the body will contain a more specific explanation of the error situation.

Error codes

Error message format

The error messages that may be returned for requests made to the Capacity Management, Provisioning and SIP Domain APIs are in JSON and have the following format:

- **`error_code`**: the error code for this error. Error codes are hierarchical: for example 100:1 is a subclass of error 100, and 100:1:2 is a subclass of error 100:1.
- **`message`**: Plain text description of the error.
- **`details`**: Informal information about the error to aid debugging. Note that the information provided may vary and there is no guarantee that the information will always be provided.

```
HTTP [HTTP error code]

{

        "details": {[informal information to aid debugging]},

        "error_code": "[API error code class]:[API error code subclass]",

        "message": "[message]"

}
```

List of all error messages

The following is a list of all error messages for the Capacity Management, Provisioning and SIP Domain APIs:

Message	HTTP Error Code	API Error Code	Description
[Depends on instance]	[Depends on instance.]	100	This class of error codes is used when there is an HTTP serving error. The message and error code depend on the type of error. The HTTP error code may be reflected in the error subclass.
An object specified in the request cannot be found by the API.	404	10	Used when an object cannot be found because the client requested the wrong object, not because of a configuration or internal error.
An object could not be matched internally due to a configuration error.	500	15	Used when an object cannot be found because of a configuration error. For example, when an object referenced from the requested object has been removed.
A validation error in the request prevented further processing.	400	20	Used when a request caused an API validation error, for example when the request contains a value that is not supported by the API or an invalid field length.
A logical error in the request prevented further processing.	400	30	Used when a request is logically incorrect, violating the current state of the system, for example a request containing quality settings that are not supported for the conference bridge type specified.
The system has reached a limit, which prevented further processing.	507	40	Used when a system limit is reached, for example when the number of records requested is higher than the supported limit.
A syntactical error in the request prevented further processing.	400	50	Used when a request caused a syntactical error, for example when the JSON syntax in the request is incorrect.
An unexpected API error occurred.	500	60	Used when there is an unexpected error generated by the Capacity Management, Provisioning or SIP Domain API.
The API generated an unexpected exception that may have left the database in a corrupt or inconsistent state.	500	60:10	Used when there is an unexpected error generated by the Capacity Management, Provisioning or SIP Domain API that may left the database in a corrupt or inconsistent state.
The API is temporarily unavailable.	503	70	Used when the API or a component it tries to access is temporarily unavailable.

Example error messages

Below are some examples of error messages:

```
HTTP 404

{

        "details": {},

        "error_code": "100:404",

        "message": "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again."

}
```

```
HTTP 405
{
        "details": {},
        "error_code": "100:405",
        "message": "The method is not allowed for the requested URL."
}

HTTP 400
{
        "details": {
                "validation_errors": {
                        "": "Unrecognized keys in mapping: \"{u'dial_out_participants': [], u'conferenc
type': u'meeting'}\""
                }
        },
        "error_code": "20",
        "message": "A validation error in the request prevented further processing."
}

HTTP 400
{
        "details": {
                "info": "Expecting property name: line 11 column 1 (char 314)"
        },
        "error_code": "50",
        "message": "A syntactical error in the request prevented further processing."
}
```

Other REST APIs

Aside from the Capacity Management, Provisioning and SIP domain APIs, TelePresence Conductor supports the following REST APIs:

- **`https://<IP address of the Conductor>/status`** - summary information about the version and status (active/inactive) of the TelePresence Conductor system.
- **`https://<IP address of the Conductor>/systemunit.xml`** - summary information about the system version.

Note: if **Automatic discovery protection** has been set to *On* on the TelePresence Conductor's **System administration** page, a username and password are required to access this information.

- **`https://<IP address of the Conductor>/api/external/`** - summary information about the configuration and status of the TelePresence Conductor system. This is currently the only supported REST API providing serviceability information.

The api/external REST API exposes tables consisting of one or more records - each with one or more fields. Most tables use a uuid as the primary key.

The following tables are supported:

- configuration/conferencefactory/pools
- configuration/conferencefactory/servicepreferences
- configuration/time
- status/cluster
- status/clusterpeer
- status/networkinterface
- status/conferencefactory/mcustatus
- status/conferencefactory/mcucallsignallingloadstatus
- status/conferencefactory/mculicenceloadstatus

Access to any undocumented resource or any resource not under **`https://<IP address of the Conductor>/status`**, **`https://<IP address of the Conductor>/systemunit.xml`** or **`https://<IP address of the Conductor>/api/external/`** is unsupported - and access may be modified or completely withdrawn without notice in a future release.

The two main groups of information are:

- Status information (read-only) : **`https://<IP address of the Conductor>/api/external/status`**
- Configuration (read-only) : **`https://<IP address of the Conductor>/api/external/configuration`**

Current status resources available include:

- **`https://<IP address of the Conductor>/api/external/status/conferencefactory/primaryconferences`** - summary information about currently running conferences
- **`https://<IP address of the Conductor>/api/external/status/alarm`** - information about all alarms, whether they're raised or lowered, and an English language translation of the descriptions of the alarms.
- **`https://<IP address of the Conductor>/api/external/status/system`** - detailed information about the system software version

The following configuration resources are available:

- **`https://<IP address of the Conductor>/api/external/configuration/dnsserver`** - DNS server configuration
- **`https://<IP address of the Conductor>/api/external/configuration/ntpserver`** - NTP server configuration

- **`https://<IP address of the Conductor>/api/external/configuration/snmp`** - SNMP server configuration
- **`https://<IP address of the Conductor>/api/external/configuration/dns`** - DNS server configuration
- **`https://<IP address of the Conductor>/api/external/configuration/conferencefactory/mcuinfo`** - used by TMS to gain information about TelePresence MCU and TelePresence Server systems known to TelePresence Conductor
- **`https://<IP address of the Conductor>/api/external/configuration/conferencefactory/mcuaddress`** - also used by TMS to gain information about TelePresence MCU and TelePresence Server systems known to TelePresence Conductor

The REST API returns results in XML format and permits the restriction of results to the addressed peer's result set by use of the **`peer=local`** query string parameter (e.g. **`https://<IP address of the Conductor>/api/external/status/networkinterface?peer=local`**)

System information

The version of the TelePresence Conductor software that is running, can be obtained using **`systemunit.xml`**.

The following is example XML for **`systemunit.xml`**:

```
<SystemUnit>
        <Name>TestConductor</Name>
        <Software>
                <Version>XC2.2</Version>
        </Software>
</SystemUnit>
```

REST API and Clusters

Some tables, containing global configuration or status information applicable to all peers in a cluster, are shared by all members of a cluster.

Other tables, containing system specific configuration or status information, contain a sub-table per cluster peer. All peers (conceptually) have access to all tables (including the system-specific tables of other peers) although by convention one cluster peer will never modify another cluster peer's system specific table.

Reading records

Reading all records

GET from **`http://<IP address of the Conductor>/api/external/<basepath>/`**

Reading some records

GET from **`http://<IP address of the Conductor>/api/external/<basepath>/<key>/<value>`**

where:

- **<key>** is the column name. The column must be indexed
- **<value>** is the value to match.

This works with explicitly indexed tables only.

To restrict the result set to a single peer, include **peer=<IP>** in the query string, where **<IP>** is the IP address of the peer to retrieve results for. This IP address must match the corresponding cluster alternate IP for the node. As a special case, substitute "local" in place of the IP address to retrieve results for the local peer only.

Pagination of results may be achieved by specifying an offset and limit in the query string. Offset is 0-based (i.e. to obtain the first record, provide a query string of "offset=0&limit=1"). For example, to obtain the second 10 results, the query string would contain "offset=10&limit=10". This may be used in conjunction with restricting results to a single peer.

When paginating, it is possible to sort by columns other than the uuid. The sort column is specified by the **sortby** query parameter. This takes the column name as its value. For example, to sort by field3, specify "**sortby=field3**".

By default, the sort order is ascending. This may be specified explicitly using the **sortdirection** query parameter. The value of the **sortdirection** parameter is either "ascending" or "descending".

Result format

Data returned from the REST API is either JSON or XML encoded.

The result format may be controlled either through use of an Accept header in the request, or by including a query string with **format={json|xml}**. The use of an Accept header is preferred. If no result format control data is provided by the client, JSON will be returned. JSON is generally more compact and often faster to parse than XML. For JSON, the results of GET requests will be returned as follows:

```
[
{ "peer": <IP>,
"num_recs": 123,
"records": [ <Record>, ... ]
},
...
]
```

where:

- **<IP>** is the IP address of the peer, as a string,
- **<Record>** is a JSON object representing a record.

There may be multiple peer descriptors in the results: one per peer in the cluster.

The **num_recs** field contains the total number of results that matched the request on a peer. The total number of results across the cluster may be calculated by summing the peers' **num_recs** fields. If it was impossible to compute the number of matching records, **num_recs** will have a value of -1.

There may be fewer than **num_recs** results in the records field. This will be the case when the request has been limited for pagination.

The result of a POST request is a list of records affected by the request. Such a JSON response would look like the following:

```
[ <Record>, ... ]
```

where **<Record>** is a JSON object representing a record.

Code examples for accessing the JSON/REST API

All access to the REST API requires authentication.

TelePresence Conductor uses HTTPS with standards-based basic HTTP authentication to restrict access to the API.

Currently, the TelePresence Conductor supports only a single username ("admin") and password - shared with the main TelePresence Conductor web UI. This most definitely will change in a future release (to allow for API-only accounts) - so all systems integrating against the TelePresence Conductor *must* allow both username and password credentials to be configurable. Do not assume that there will always be an "admin" account.

Reading values from the REST API is easy. The examples below assume the existence of a user named "admin" with a password of "xxx":

Linux curl (JSON results from public API)

```
curl --user admin:xxx https://<IP address of the Conductor>/api/external/status/alarm
```

Linux curl (XML results from public API)

```
curl --user admin:xxx  -H "Accept: application/xml" https://<IP address of the
Conductor>/api/external/status/alarm
```

Linux wget (JSON results from a public API):

```
wget --no-check-certificate --http-user admin --http-password xxx https://<IP address of
the Conductor>/api/external/status/alarm
```

python:

```
import urllib2

theurl = 'https://<IP address of the Conductor>/api/external/status/alarm'

username = 'admin'

password = 'xxx'

passman = urllib2.HTTPPasswordMgrWithDefaultRealm()

passman.add_password(None, theurl, username, password)

authhandler = urllib2.HTTPBasicAuthHandler(passman)

opener = urllib2.build_opener(authhandler)

urllib2.install_opener(opener)

pagehandle = urllib2.urlopen(theurl)

pagehandle.read()
```

Discovering the version of the TelePresence Conductor

Before accessing the TelePresence Conductor REST or XML-RPC API, external systems should check the version of the TelePresence Conductor software by calling **`device.query`** in order to adjust their behavior (if need be) to be appropriate to the version of TelePresence Conductor they're accessing.

Other APIs

SNMP

The TelePresence Conductor has limited support for SNMP. To view the details:

1. Enable SNMP (for example, by selecting *v2c* on the **System > SNMP** page)
2. Enter the command **`snmpwalk`** from a Linux workstation to "explore":
   `snmpwalk -c public -v2c <IP address of TelePresence Conductor`

Syslog log messages

Listed below are all the messages that can appear in the TelePresence Conductor's event logs, and appear in the remote syslog feed if enabled on the TelePresence Conductor, together with their parameters.

Event log messages with a severity of "Info"

Message	Parameters
Successful login into the interactive debugging console.	Username
A conference has been deleted.	Conference_bridge_conference_name, Conference_name, Conference_bridge_UUID, Conference_bridge_address, Conference_unique_identifier
Cannot allocate conference bridge resource.	Tag, Destination-alias, Source_protocol, Source_registered_alias,Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address,Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_unique_identifier
The maximum number of participants for this role has been reached.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_unique_identifier
A request to join a conference was successfully processed.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_bridge_UUID, Conference bridge uuid, Conference_bridge_address, Conference_bridge_type, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Process_time, Request_unique_identifier, Pre-configured_endpoint
A request has been received from a client for a participant to create or join a conference.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Request_unique_identifier
Attempting to add an auto-dialed participant to a conference.	Auto-dialed_participant_name, Conference_name, Conference_template_name

Message	Parameters
A conference bridge that was previously unusable is now active.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference bridge type, Detail, Conference_name, Conference_bridge_conference_name, Status_detail
An attempt has been made to access the interactive debugging console.	Username
An incoming call request has been rejected because the conference is not present.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases,Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail, Reason, Conference_name, Conference_bridge_conference_name,Conference_unique_identifier, Conference_bridge_participant
An API request has been received.	URI, HTTP_method, Arguments, HTTP_client_protocol
Recovered from a network partition; service is resumed.	
A conference has been marked for deletion.	Conference_bridge_conference_name, Conference_name, Conference_bridge_UUID, Conference_bridge_address, Conference unique identifier
A management request has been received.	Command, Conference_name, Conference_bridge_conference_name, Participant_name, Participant_type, Participant_protocol, Requester_(VCS/Unified_CM/client)_address)
A conference has been successfully created on the conference bridge and is ready to receive participants.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases,Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Process_time, Request_unique_identifier
A conference's cascade has been marked for deletion.	Conference_bridge_conference_name, Conference_name, Conference_bridge_UUID, Conference_bridge_address, Cascade_UUID, Conference unique identifier
The TelePresence Conductor application has been stopped.	Detail
The clustering system partition status has been set.	Old_partition_state, New_partition_state
Request rejected. The conference alias has an invalid role for the conference type.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_alias, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Conference_bridge_UUID, Conference bridge uuid, Conference_bridge_address,Conference bridge type, Status_good, Config_good, Reason, Conference_name, Conference_bridge_conference_name, Conference_template_name, Participant_role, Conference_unique_identifier

Message	Parameters
The TelePresence Conductor application has started.	
A conference duration has expired. TelePresence Conductor will now terminate the conference.	Conference_name, Conference_bridge_UUID
A conference has been successfully cascaded to an additional MCU.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases,Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_bridge_UUID, Conference bridge uuid, Conference_bridge_address, Conference bridge type, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Request_unique_identifier, H323_cascade_call_routing
A conference's cascade has been deleted.	Conference_bridge_conference_name, Conference_name, Conference_bridge_UUID, Conference_bridge_address, Cascade_UUID, Conference_unique_identifier
An incoming call match string has been modified to include the SIP domain supplied by the API instead of the local domain.	Original_match_string, New_match_string

Event log messages with a severity of "Debug"

Message	Parameters
A participant's allocated resource has been reduced.	Participant_name, Resource_change
The TelePresence Conductor application has been stopped.	Detail
A participant's allocated resource has been increased.	Participant_name, Resource_change

Event log messages with a severity of "Warning"

Message	Parameters
An error occurred while communicating externally.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail, Conference_name, Conference_bridge_conference_name, Command
A CPL 'reject' has been returned to a client. Reasons could include: no matching alias, no conference template, or no dial plan prefix configured.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Detail, Request_unique_identifier, Conference_bridge_participant

Message	Parameters
An auto-dialed participant has failed to be added to a conference. This is because H.323 participants are not supported in Unified CM deployments.	Auto-dialed_participant_name, Conference_name, Conference_template_name
A Location has been configured with an unknown IP address.	Location, IP_address
Auto-dialed participant address cannot be resolved.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases,Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Conference_template_UUID, Conference_name,Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Auto-dial_participant_match, Auto-dial_participant_rule, Auto-dial_participant_address, Auto-dial_participant_keep_alive, Auto-dial_participant_protocol, Auto-dial_participant_role
A conference bridge is 'Unusable' because it is clustered and this TelePresence Conductor does not support clustered conference bridges.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Status_detail
An error occurred in the controller tick; if the problem persists, reboot the system.	
More chairpersons than expected found on conference.	Conference_name, Conference_bridge_conference_name, Chairperson_count, Reserved_chairperson_participants
Conference bridge pool resource usage is approaching full capacity.	Conference_bridge_pool_UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID
Application watchdog has detected a jump in system time. Check your NTP settings.	Last_modified, Current_time
A conference has been destroyed because it is missing from an MCU.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_name, Conference_bridge_conference_name
A conference bridge that was previously active is now unusable.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference bridge uuid, Conference_bridge_address, Conference bridge type, Detail, Conference_name, Conference_bridge_conference_name
A CPL request with no alias specified, or with an alias that does not use UTF-8, has been received from a client.	

Message	Parameters
Changes to vital conference bridge configuration have been detected. All conferences on this conference bridge have been deleted from the database.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conferences
Request rejected. The conference exists but is not responding in a timely fashion.	Tag, Destination-alias,Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail, Status_good, Config_good, Reason, Conference_name, Conference_bridge_conference_name, Timeout
Unable to process request because the requested service was not found.	
A conference or a cascade could not be recovered.	Conference_name, Conference_bridge_conference_name
Keep conference alive will be ignored for auto-dialed participant.	Conference_template_name, Auto-dialed_participant_name
An attempt to create a conference or a cascade was unsuccessful.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Request_unique_identifier
An error occurred while communicating externally, retrying.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail, Conference_name, Conference_bridge_conference_name, Command
Cannot resolve conference bridge hostname.	Conference_bridge_address
Not enough conference bridge resource to handle request.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_unique_identifier
Conference bridge pool resource usage has reached full capacity.	Conference bridge pool UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID
A conference has been destroyed because its conference bridge is not configured.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_name, Conference_bridge_conference_name
A conference bridge is reporting zero available resource.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type

Message	Parameters
Adding an auto-dialed participant to a conference bridge failed.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Auto-dial_participant_match, Auto-dial_participant_rule, Auto-dial_participant_address, Auto-dial_participant_keep_alive, Auto-dial_participant_protocol, Auto-dial_participant_role
A Location has been configured without an ad hoc IP address.	Location
An API request could not be processed.	Command, Conference_name, Conference_bridge_conference_name, Participant_name, Participant_type, Participant_protocol, Detail, Xmlrpc_parameters
An API request was made to create a conference using a non-scheduled conference template; the factory.conferencecreate request may fail if the conference has already started.	Conference_template_name
A cascade link has been lost. The system will attempt to rebuild it.	Conference_name, Conference_bridge_conference_name, Primary_conference_bridge_UUID, Primary_conference_bridge_address, Cascade_bridge_UUID, Cascade_bridge_address
Deleting from a conference bridge a conference unknown to this peer.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail,Conference_name,Conference_bridge_conference_name
A conference bridge is unusable because it is not in a running state.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type
Incoming call request rejected.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail, Reason, Conference_name, Conference_bridge_conference_name, Conference_unique_identifier, Pre-configured_endpoint
Auto-dialed participant request has been sent to the conference bridge.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Auto-dial_participant_match, Auto-dial_participant_rule, Auto-dial_participant_address, Auto-dial_participant_keep_alive, Auto-dial_participant_protocol, Auto-dial_participant_role
A conference bridge is taking a long time to respond to requests.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail

Message	Parameters
A WebEx-enabled conference was created without content enabled on the conference template; this will prevent video callers seeing or presenting any WebEx shared content.	Conference_template_name
A Location with an ad hoc IP address is missing an ad hoc conference template.	Location
A conference has been removed from the database because it is missing from a conference bridge.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_name, Conference_bridge_conference_name, Conference_part_type
A conference was not recovered because the conference template was not found.	Conference_template_UUID, Conference_name, Conference_bridge_uuid
Failure to log into the interactive debugging console.	Username

Event log messages with a severity of "Error"

Message	Parameters
A TelePresence MCU pool is erroneously in a TelePresence Server Service Preference.	Conference_bridge_pool_UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID
A prohibited JSON key has been provided via the conference template.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier
Cannot create a conference, because this version of conference bridge requires both Chair PIN and Guest PIN to be set when using PINs.	Conference_template_name, Conference_bridge_UUID, Conference_name, Conference_bridge_conference_name, Conference_bridge_address
Bad conference name.	Call_policy_prefix, Match
A conference has been destroyed because it is missing from an MCU.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_name, Conference_bridge_conference_name
A conference bridge is missing a dial plan prefix. This is causing conferences to fail.	Conference_bridge_name
Unable to find parent record.	Table, Record, Field, Parent_uuid

Message	Parameters
An attempt to create a conference or a cascade was unsuccessful.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier, Request_unique_identifier
Request rejected. The TelePresence Conductor service is currently unavailable.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail, Status_good, Config_good, Reason, Conference_name, Conference_bridge_conference_name
Invalid JSON has been provided via the conference template.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Incoming_alias_match, Conference_name_rule, Detail, Conference_alias_UUID, Conference_alias_name, Conference_alias_description, Participant_type, Participant_role, Conference_template_UUID, Conference_name, Conference_bridge_conference_name, Conference_template_name, Conference_unique_identifier
Application watchdog has performed a number of system restarts but the application has not responded. Manual intervention is required.	
An error occurred while communicating with the internal database.	Communication_type
An attempt to increase the resource allocated to a participant has failed.	Participant_name, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_name, Conference_bridge_conference_name
Release key could not be found.	
An API request has failed.	Fault_code, Fault_message, HTTP_method, Request_path
A pool has a conference bridge type that is different from its associated Service Preference.	Service_Preference_uuid, Service_Preference_name, Service_Preference_conference_bridge_type, Service_Preference_pool_uuids, Conference_bridge_pool_UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID
TelePresence Conductor application is not responding. Restarting system.	
An active conference bridge pool does not contain any conference bridges.	Conference_bridge_pool_UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID

Message	Parameters
Incoming call request rejected.	Tag, Destination-alias, Source_protocol, Source_registered_alias, Source_address, Unauthenticated_source_aliases, Requester_(VCS/Unified_CM/client)_address, Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Detail, Reason, Conference_name, Conference_bridge_conference_name, Conference_unique_identifier, Pre-configured_endpoint
The TelePresence Conductor application has been stopped.	Detail
A conference bridge has a conference bridge type that is different from the conference bridge type of the pool to which it belongs.	Conference_bridge_UUID, Conference_bridge_uuid, Conference_bridge_address, Conference_bridge_type, Conference_bridge_pool_UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID
A conference bridge does not have a Location of type 'rendezvous' or 'both'.	Conference_bridge_pool_UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID
Invalid release key detected.	
A conference bridge does not belong to a conference bridge pool.	Conference_bridge_name
A TelePresence Server pool is erroneously in a TelePresence MCU Service Preference.	Conference_bridge_pool_UUID, Conference_bridge_pool_name, Conference_bridge_pool_type, Service_Preference_ID
Conflict between conference alias and conference bridge dial plan prefix.	Match, Conference_bridge_prefix
A Location without a trunk IP address is being referenced by a conference bridge pool.	Location_name
A lecture-type conference template is not referenced by an alias with a 'chairperson' or 'guest' role.	Conference_template_name, Missing_alias_role
A conference template is not referenced by any Service Preferences.	Parent_name

Assured scheduling

For API clients to be able to offer assured scheduling of conferences managed by TelePresence Conductor, the TelePresence Conductor API provides the following features:

- Capacity Management API [p.27]
  The main purpose of the Capacity Management API is to provide the API client with the capacity information for a conference to be able to create scheduled conferences and to be able to offer assured scheduling to its users.
- factoryResourceLimits [p.11] struct within **`factory.conferencecreate`**
  The parameter factoryResourceLimits limits the resources for a conference and helps to prevent arbitrary conference enlargement.

The following restrictions apply to assured scheduling:

- We recommend that you use the TelePresence Conductor in back-to-back user agent (B2BUA) mode. If the TelePresence Conductor is used with the Cisco VCS's external policy server interface, the TelePresence Conductor will initially reserve resources for the number of screens defined on the conference template and then optimize resources back to the actual number of screens the endpoints are capable of.
- We recommend that you do not use cascading. Assured conferences are only supported where a Service Preference contains a single conference bridge. If there is more than one conference bridge in a pool the capacity information provided is 'best effort'.
- Pre-configured endpoints defined on the TelePresence Conductor are not considered in the capacity calculations and are therefore not supported in assured scheduling.
- Assured scheduling only supports a single pool in a Service Preference and the capacity information returned represents the sum of the capacities of all the conference bridges in the pool.
- Dedicated content ports on the TelePresence MCU are not considered in the capacity calculations.
- For assured scheduling, the number of chairperson participants to reserve should be set to 1. The Capacity Management API will assume this value and will not include it in the **`ServiceParams`** data structure.
- The Cisco TelePresence T3 multiscreen endpoint is not supported and assured scheduling may not handle the T3 correctly.

Examples

The following examples demonstrate how the costs for different conferences are calculated and what the resulting API request parameters are.

Conference hosted on a TelePresence Server

In this example the conference that the endpoints dial into supports:

- 1080p 30fps video
- multi-channel audio
- 720p 30fps content

Given the quality settings above, the Capacity Management API would return the following service costs for a single TelePresence Server:

- MaxCalls = 200 (104 when running TelePresence Server version 3.1 or earlier)
- MaxCalls per conference = 104
- Media = 30720
- Licenses = 30720
- Cost for this Alias:
  Per conference cost
  - {Calls=0, Media=0, Licenses=0}

  Total participant cost
  - 1 screen {Calls=1, Media=3072, Licenses=3840}
  - 2 screen {Calls=1, Media=4992, Licenses=5040}
  - 3 screen {Calls=1, Media=6912, Licenses=7560}
  - 4 screen {Calls=1, Media=8832, Licenses=10080}

To work out the cost of the conference, all number of calls, media and license requirements are added up. The table below provides examples for different numbers and types of endpoints:

Endpoints in the conference	Per conference cost	Total participant cost	API request parameters
4 x single-screen endpoints	{Calls=0, Media=0, Licenses=0}	{Calls=4, Media=12288, Licenses=15360}	factoryResourceLimits:{ "signalling": 4, "media": 12288, "licenses": 15360}
2 x 3-screen TIP endpoints	{Calls=0, Media=0, Licenses=0}	{Calls=2, Media=17664, Licenses=20160}	factoryResourceLimits:{ "signalling": 2, "media": 17664, "licenses": 20160}
4 x single-screen endpoints + 2 x 3-screen TIP endpoints	{Calls=0, Media=0, Licenses=0}	{Calls=6, Media=29952, Licenses=35520	factoryResourceLimits:{ "signalling": 6, "media": 29952, "licenses": 35520}

Conference supporting content hosted on a TelePresence MCU

In this example the conference that the endpoints dial into supports content and runs on a TelePresence MCU.

The Capacity Management API would return the following service costs for a single TelePresence MCU:

- MaxCalls = 80
- MaxCalls per conference = 80
- Media = not applicable
- Licenses = not applicable
- Cost for this Alias:
  Per conference cost
  - {ports=1}

  Total participant cost
  - 1 screen {ports=1}

Note that the per conference cost is 1 because the conference supports content.

To work out the cost of the conference, the ports are added up:

Endpoints in the conference	Per conference cost	Total participant cost	API request parameters
4 x single-screen endpoints	{ports=1}	{ports=4}	factoryResourceLimits:{ "ports": 5}

Conference not supporting content hosted on a TelePresence MCU

In this example the conference that the endpoints dial into does not support content and runs on a TelePresence MCU.

The Capacity Management API would return the following service costs for a single TelePresence MCU:

- MaxCalls = 80
- MaxCalls per conference = 80
- Media = not applicable
- Licenses = not applicable
- Cost for this Alias:
  Per conference cost
  - {ports=0}

  Total participant cost
  - 1 screen {ports=1}

Note that the per conference cost is 0 because the conference does not support content.

To work out the cost of the conference, the ports are added up:

Endpoints in the conference	Per conference cost	Total participant cost	API request parameters
4 x single-screen endpoints	{ports=0}	{ports=4}	factoryResourceLimits:{ "ports": 4}

API performance and security

Current API performance limits

The currently supported API performance limits are:

Feature	Limit
Maximum number of API clients	<= 4
Maximum number of API connections per client	1
Maximum number of concurrent API client connections (across all clients, including TMS, PCM and your application)	<= 4

API performance considerations

Accessing the REST API of the TelePresence Conductor often requires the TelePresence Conductor to lock access to the database and wait for database replication to complete in order to ensure that a coherent response is returned. This could adversely affect performance of an entire TelePresence Conductor cluster. Accessing the XML-RPC API of the TelePresence Conductor may cause the TelePresence Conductor to contact the conference bridge(s) - thus causing load on the conference bridge as well as on the TelePresence Conductor.

Repetitive or high volume accesses to the TelePresence Conductor API may therefore adversely affect the call handling performance of the TelePresence Conductor and the conference bridges managed by the TelePresence Conductor - and therefore designs, which require this, should be avoided.

- Accessing the REST or XML RPC API incurs a performance penalty on the TelePresence Conductor. If the TelePresence Conductor is under heavy load, API responses may be delayed.
- Many XML RPC API invocations are simply passed on to one or more underlying conference bridges- thus causing additional load and performance penalties on the conference bridges themselves. Other calls involve TelePresence Conductor accessing its own database, incurring cluster-wide REST access penalties.
- Some types of conference bridges have a very limited capacity for processing simultaneous XML RPC requests - so TelePresence Conductor is forced to serialize XML RPC requests for those conference bridges in order to avoid overloading them. This can result in slow response times even if the TelePresence Conductor itself is not under heavy load.
- Finally, the CPU time cost of doing the handshakes required for establishment of an HTTPS connection to TelePresence Conductor is quite high.

So, to ensure good performance:

- Please remember that various systems other than yours (such as Cisco TMS and Cisco Prime Collaboration Manager) may also be simultaneously accessing the TelePresence Conductor API
- Please limit the number of concurrent TCP connections you make, as the TelePresence Conductor can handle no more than four concurrent API connections without degrading system performance.
- Please remember that the TelePresence Conductor may be under heavy call handling load. Because call handling may be just as important (or even more important) than handling the API request, use API requests sparingly.

- Please avoid performing large amounts of background polling (using REST or XML-RPC) for information that is not needed
- Please avoid making multiple concurrent API requests to the TelePresence Conductor - wait for your last request to complete before making the next one.
- In cases where a client will be making multiple REST and/or XML-RPC requests to TelePresence Conductor in quick succession, please make use of HTTP 1.1 connection keep-alives, to allow multiple requests to be sequentially handled using a single connection to TelePresence Conductor. This will avoid the considerable expense of dropping/re-establishing an HTTPS connection for every request.

The best architecture for a client of the TelePresence Conductor API 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 from being impaired by unmanageable numbers of API requests.

Security considerations

For security reasons, the TelePresence Conductor API should only be accessed over HTTPS. Production deployments of TelePresence Conductor should install a valid certificate, signed by an appropriate certificate authority. Clients of the TelePresence Conductor API should then (optionally) allow the administrator to configure the clients such that the client service checks the validity of the certificate of the TelePresence Conductor that they are connecting to (and thus protect the overall service from man-in-the-middle attacks).

Be aware: if any TelePresence MCU or TelePresence Server configured on the TelePresence Conductor is configured to use HTTP rather than HTTPS, then the TelePresence Conductor will transmit sensitive information in the clear to the XML-RPC API of the conference bridge(s). In security sensitive deployments it is important for solution security to also enable HTTPS on the conference bridges and to configure the TelePresence Conductor to communicate to the conference bridges over HTTPS.

Appendix A: System limits

TelePresence Conductor performance/API goals and limits

General TelePresence Conductor limits

Feature	Limit
Concurrent conferences	<= 1,000
Conference bridges	<= 30
Total number of calls	<= 2,400
Maximum number of participants per conference	<= 2,342

XML-RPC API limits

Feature	Limit
Conference aliases	<= 1,000
Conference templates	<= 1,000
Auto-dialed participants	<= 1,000
Conference create events per second	<= 2
Conference join events per second	<= 8

Provisioning API limits

Feature	Limit
Conference bundles	100,000
Direct match aliases	10 per ConfBundle, 200,000 in total
Auto-dialed participants	10 per ConfBundle, 100,000 in total

Monitoring/Management API performance goals

Feature	Limit
Monitored conferences	<= 40
Minimum conference poll period	>= 5 seconds
Maximum poll requests per second	<= 8
Mute requests per second	<= 10

The above limits are not enforced - but should not be exceeded in normal usage. If your monitoring/management system can safely be designed to make poll requests less frequently than the maximum supported rate(s) then that is better for overall system performance. We aim to test to 150% of each limit to ensure we have enough performance "headroom".

We strongly suggest that such performance/capacity testing is appropriate for customers of the TelePresence Conductor API too - as it might be that the burden placed on the TelePresence Conductor by external API clients is significant.

In practice, this means you should arrange to test the impact of your system running against a heavily loaded TelePresence Conductor - and verify that the presence of your system does not prevent TelePresence Conductor from meeting its performance targets.