



Cisco TelePresence Conductor

Product Programming Reference Guide

XC4.0

August 2015

Contents

Introduction	5
Current TelePresence Conductor API clients	5
Changes in version XC4.0	5
Remote Management XML-RPC API	7
Authentication	7
XML-RPC return values	8
conference.create	8
Input parameters	8
Returned data	9
conference.destroy	9
Input parameters	9
Returned data	9
conference.end	10
conference.enumerate	10
Input parameters	10
Returned data	10
conference.modify	11
Input parameters	11
Returned data	12
device.network.query	12
Input parameters	12
Returned data	12
device.query	13
Input parameters	13
Returned data	13
factory.conferencecreate	13
Input parameters	13
Returned data	17
factory.conferencemodify	18
Input parameters	18
Returned data	20
factory.health.query	20
Returned data	21
factory.webex.add	21
Input parameters	21
Returned data	22
feedbackReceiver	22
feedbackReceiver.extended	22
Feedback Events	24
feedbackReceiver.extended.configure	24
feedbackReceiver.extended.query	25
feedbackReceiver.extended.remove	26
Participant	26
participant.add	27
participant.diagnostics	28
participant.disconnect	28
participant.enumerate	29
participant.message	31

participant.modify	32
participant.remove	34
Fault codes	34
REST APIs	37
Capacity Management API	37
Data structures	37
Resources object	38
ParticipantCost object	39
ServiceParams object	41
ServiceCost operation	42
Provisioning API	43
Data structures	44
QualityInfo object	44
ServiceInfo object	46
ConfBundleID object	47
ConfBundleVer object	48
ConfBundle object	48
Alias object	55
AutoDialedParticipant object	56
Role determined by PIN	57
ServiceInfo operation	58
QualityInfo operation	58
ConfBundles operation	58
SIP Domain API	60
SIPDomainSpec object	60
SIPDomainSpec operation	61
Owners API	61
Owners object	61
Owner object	62
Owners operation	63
Error codes	63
Error message format	63
List of all error messages	63
Example error messages	64
Other REST APIs	65
External REST API	66
Multipartylicense REST API	67
System information	67
REST API and Clusters	68
Reading records	68
Result format	68
Code examples for accessing the JSON/REST API	69
Scheduling via Cisco TMS	71
Examples	71
Conference hosted on a TelePresence Server	72
Conference that supports content hosted on a TelePresence MCU	72
Conference that does not support content hosted on a TelePresence MCU	73
SNMP	74

API performance and security	75
Current API performance limits	75
API performance considerations	75
Security considerations	76
Appendix A: System limits	77
TelePresence Conductor performance/API goals and limits	77
General TelePresence Conductor limits	77
XML-RPC API limits	77
Provisioning API limits	77
Monitoring/Management API performance goals	77

Introduction

This document describes the set of application programming interfaces (APIs) that is available on the Cisco TelePresence Conductor version XC4.0.

There are plans to update the APIs in future releases to better provide a generic API independent of the conference bridges or other resources that the TelePresence Conductor may be managing. These documented APIs are available in version XC4.0, but may change in future releases.

Many of the XC4.0 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

Changes in version XC4.0

The following changes have been introduced in Cisco TelePresence Conductor API version XC4.0:

Support for the `conference.create` struct `ownerID`

The TelePresence Conductor now supports the struct `ownerID`, if it is provided as an optional parameter within a `conference.create` call. A Unified CM running version 10.5 or later may supply this parameter when creating an ad hoc conference on TelePresence Conductor. TelePresence Conductor uses the information provided as part of the `ownerID` struct to work out what type of license the conference initiator has got.

Support for the `factory.conferencecreate` String `factoryOwnerId`

The TelePresence Conductor now supports the String `factoryOwnerId`, if it is provided as an optional parameter within a `factory.conferencecreate` call. TelePresence Conductor uses `factoryOwnerId`, which is the identifier of the conference initiator, to work out what type of license the conference initiator has got.

Addition of new `ConfBundle` attributes

The `ConfBundle` object has two new attributes: `owner_id` and `external_owner_id`. They are both GUIDs that identify the owner of the `ConfBundle` object. The attributes of the `ConfBundle` define the type of multiparty license the owner has.

- `owner_id` is used to link a `ConfBundle` object to an `<Owner>` object.
- `external_owner_id` is used to link the conference initiator ID passed to TelePresence Conductor via a `conference.create` or `factory.conferencecreate` method to this `ConfBundle` object, thus linking it to the multiparty license of the conference initiator.
- `multi_party_license` is used to define the type of license the conference initiator has - either personal multiparty (PMP) or shared multiparty (SMP).

Addition of a new Owners API

The **Owners** API has an **Owners** object defined that contains an array of **Owner** objects. The **Owner** object is a subset of the **ConfBundle** object. It contains information about the initiator of a conference, including his or her multiparty license.

The TelePresence Conductor supports PUT and DELETE methods for the **Owners** API.

Addition of the multipartylicense REST API

The TelePresence Conductor now supports a GET method for the new REST API **https://<IP address of the Conductor>/api/4.0/multipartylicense**. It provides summary information about the number of multiparty licenses that have been added as an option key to the TelePresence Conductor. The totals apply across all peers of a cluster.

Addition of a new value for bridge_capabilities used in the ServiceInfo object

The TelePresence Conductor now supports the new value *centralized_licensing* to be included in the **bridge_capabilities** Array used in the **ServiceInfo** object. If *centralized_licensing* is included in the Array, TelePresence Conductor informs the API client that it can provide owner information via its **ConfBundle** PUT operations.

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 the 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.end](#)
- [conference.enumerate](#)
- [conference.modify](#)
- [device.network.query](#)
- [device.query](#)
- [factory.conferencecreate](#)
- [factory.conferencemodify](#)
- [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: 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
<code>authenticationUser</code>	String	Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.

Parameter	Type	Description
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.create

This call creates a new conference via TelePresence Conductor. It is deprecated. Use [factory.conferencecreate](#) instead.

Input parameters

Required inputs

Parameters	Type	Description
conferenceName	String	The name of the conference to create.

Optional inputs

Parameters	Type	Description
durationSeconds	Integer	The period of time, in seconds, for which the conference will be active. A value of 0 means that the conference duration is unlimited.
reservedVideoPorts	Integer	The number of video ports to reserve for the conference.
encryption	String	The type of encryption for this conference. One of the following values: <ul style="list-style-type: none"> ■ optional ■ required ■ forbidden

Parameters	Type	Description
ownerID	struct	<p>A struct that contains information about the initiator of this conference. The information allows the TelePresence Conductor to work out which license the initiator of the conference has. The TelePresence Conductor looks through the ConfBundle objects first and then through the <Owner> objects to find a matching external_owner_id.</p> <p>This struct may be supplied by Unified CM when creating an ad hoc conference.</p> <p>If userObjectGUID is provided, it is used as the owner ID. If mobileUserObjectGUID is provided, it is used as the owner ID. If neither attribute is provided, the conference is created without an owner ID.</p> <p>If a String is passed to the TelePresence Conductor, instead of an ownerID struct, the TelePresence Conductor returns a type error.</p>
→ objectGUID	String	An optional identifier for a conference initiator. TelePresence Conductor uses this as the owner ID (ID of the conference initiator), if provided.
→ mobileUserObjectGUID	String	An optional identifier for a conference initiator. If userObjectGUID is not provided and mobileUserObjectGUID is provided, TelePresence Conductor uses this as the owner ID (ID of the conference initiator).

Returned data

Parameters	Type	Description
status	String	<p>On success: "operation successful".</p> <p>On failure: "error".</p>

conference.destroy

This call disconnects all participants in the conference and ends the conference. It is used to end Unified CM 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.end

This call has the same functionality as [conference.destroy \[p.9\]](#). It is deprecated. Use `conference.destroy` instead.

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
<code>conferences</code>	Array	Array of conference structs as defined by the TelePresence MCU API. TelePresence Conductor will add the following parameters to the structs:
→ <code>factoryTemplateType</code>	String	The type of template: <i>meet</i> , <i>lesson</i> or <i>unknown</i> . The type is <i>unknown</i> , if the configuration has changed since this conference was started.
→ <code>isCascaded</code>	Boolean	<i>true</i> if the conference is cascaded, <i>false</i> otherwise.
→ <code>factoryConferenceId</code>	String	Conference identifier supplied by the TelePresence Conductor for tracking.
→ <code>factoryWebEx</code>	String	<i>None</i> represents a conference that does not support WebEx. (This is the default value.) <i>SIP</i> represents a conference that supports a SIP WebEx conference, where resources for one call is used on the primary conference bridge. <i>SIP-TSP</i> 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.

Parameters	Type	Description
→ encryption	String	The encryption requirements for endpoints in this conference. <i>optional</i> : endpoints in this conference may have encryption enabled or disabled. (This is the default value.) <i>required</i> : endpoints in this conference must have encryption enabled. <i>forbidden</i> : endpoints in this conference must not have encryption enabled. This is only applicable to TelePresence Servers, not to TelePresence MCUs.
→ locked	Boolean	Whether the conference is locked or not. If a conference is locked, it continues to run with its existing participants, but no new participants can dial in. It is still possible for the API client to add more participants to the conference using the call participant.add .
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.

conference.modify is deprecated. Use [factory.conferencemodify](#) instead.

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 to use the customLayout or not.
newParticipantsCustomLayout	Boolean	Whether new participants will use the customLayout or not. This is only valid if customLayoutEnabled is <i>true</i> .
setAllParticipantsToCustomLayout	Boolean	<i>true</i> sets all participants to immediately see the conference custom layout. If <i>false</i> nothing happens to the layout of participants. Only valid if customLayoutEnabled is <i>true</i>

Parameters	Type	Description
customLayout	Integer	<p>The index of the video layout seen by the participant(s). This is only valid if customLayoutEnabled is <i>true</i>.</p> <p>See the relevant Cisco TelePresence MCU API reference guide for a list of available layouts and corresponding index values. For TelePresence Servers the closest approximation to the specified layout will be used.</p> <p>This value is sent to all conference bridges (primary and cascade) that are currently hosting the conference. If another cascade conference bridge is added to the conference after this conference.modify call has been made, the value is not sent to the new cascade conference bridge. In this case participants joining the new cascade conference bridge will not use the customLayout.</p>
locked	Boolean	<p>Whether the conference is locked or not.</p> <p>If a conference is locked, it continues to run with its existing participants, but no new participants can dial in. It is still possible for the API client to add more participants to the conference using the call participant.add.</p>

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	<p>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 auto-dialed participants associated with the conference template.</p> <p>The default is 5.</p>
factoryMaxDurationMinutes	Integer	<p>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.</p>
factoryOverridePIN	String	<p>If present, this is the String of numeric digits that participants with host privileges need to enter to join the conference. It overrides the PIN defined on the conference template. For a Meeting-type conference this is the PIN used for meeting participants.</p> <p>The String must be less than 32 characters.</p>
factoryOverrideGuestPIN	String	<p>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.</p> <p>The String must be less than 32 characters.</p>
webEx	String	<p><i>None</i> creates a conference that does not support WebEx. (This is the default value.)</p> <p><i>SIP</i> makes the TelePresence Conductor reserve resources for one call on the primary conference bridge.</p> <p><i>SIP-TSP</i> makes the TelePresence Conductor reserve resources for two calls on the primary conference bridge.</p>
factoryResourceLimits	resourceLimits struct	<p>The resource limits for this conference, made up of signalling, media and licences for a conference bridge of type <i>tsmcu</i> and made up of ports for a conference bridge of type <i>mcu</i>.</p>
→ signalling	Integer	<p>Limit on the number of participants/calls that this conference may use. This is only applicable to conference bridges of type <i>tsmcu</i>.</p> <p>Accepted values are 0 to 2400.</p> <p>Use 0 to create a conference with an unlimited number of participants/calls. The number of participants will be limited by the capacity of the conference bridges in this case.</p>
→ media	Integer	<p>Limit on the media resources that this conference may use. This is only applicable to conference bridges of type <i>tsmcu</i>.</p>

→ licences	Integer	Limit on the licence resources that this conference may use. This is only applicable to conference bridges of type <i>tsmcu</i> .
→ ports	Integer	<p>Limit on the number of ports that this conference may use. This is only applicable to conference bridges of type <i>mcu</i>.</p> <p>Accepted values are 0 to 2400.</p> <p>Use 0 to create a conference with an unlimited number of participants/ports. The number of participants will be limited by the capacity of the conference bridges in this case.</p>
encryption	String	<p>The encryption requirements for endpoints in this conference.</p> <p><i>optional</i>: endpoints in this conference may have encryption enabled or disabled. (This is the default value.)</p> <p><i>required</i>: endpoints in this conference must have encryption enabled.</p> <p><i>forbidden</i>: endpoints in this conference must not have encryption enabled. This is only applicable to TelePresence Servers, not to TelePresence MCUs.</p>

factoryLayout	String	<p>The layout that the conference participants will see when they join the conference. The parameter allows you to override the layout specified within the layout parameter of the ConfBundle object in the Provisioning API.</p> <p>If factoryLayout is omitted, the layout specified within the ConfBundle object is not overridden.</p> <p>The accepted values for factoryLayout are:</p> <ul style="list-style-type: none">■ <i>equal</i>: conference participants are shown in a grid pattern of equal sized panes, up to 4x4. (Not applicable to multiscreen endpoints)■ <i>active</i>: 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.■ <i>prominent</i>: 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)■ <i>single</i>: the active speaker is shown in one full-screen pane. <p>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.</p> <p>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.</p>
factoryOverrideConferenceDisplayName	String	<p>The conference display name is normally generated automatically by TelePresence Conductor. You can override the conference display name using factoryOverrideConferenceDisplayName. The String does not need to be unique.</p> <p>The conference display name may be displayed on the endpoint screen, if the underlying conference bridge has the appropriate capabilities.</p>

factoryOwnerId	String	<p>An optional identifier for the initiator of this conference. The ID allows the TelePresence Conductor to work out which license the initiator of the conference has. The TelePresence Conductor looks through the ConfBundle objects first and then through the <Owner> objects to find a matching external_owner_id.</p> <p>This String may be supplied by Cisco TMS when creating a scheduled conference on TelePresence Conductor.</p> <p>If no String is supplied, the TelePresence Conductor will assume that the conference initiator does not have a CMR and hence a ConfBundle associated with it.</p>
-----------------------	--------	---

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". On failure: "error".
info	String	This is returned when the call failed. It provides information on why the call failed.

Parameters	Type	Description
conferenceName	String	Conference name supplied by the TelePresence Conductor - for tracking messages. This is returned when the call was successful and in some failure cases. For example, if the conference already exists, the name of the existing conference is returned.
factoryConferenceId	String	Conference identifier supplied by the TelePresence Conductor for tracking. This is returned when the call was successful and in some failure cases. For example, if the conference already exists, the conference identifier of the existing conference is returned.
factory_conference_id	String	Conference identifier supplied by the TelePresence Conductor for tracking. This is returned only when the call was successful. It is retained for backwards compatibility. It has been deprecated and in the future factoryConferenceId should be used.

factory.conferencemodify

This call modifies a conference that is currently being managed by the TelePresence Conductor.

It replaces [conference.modify](#), which has been deprecated.

Input parameters

Required parameters

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

Optional parameters

Parameters	Type	Description
h239Important	Boolean	Whether the H.239 channel is set on the conference bridge to be important or not.
factoryMinDurationMinutes	Integer	The minimum time a conference will live even with no participants. The value for this parameter must not be larger than the value for factoryMaxDurationMinutes . If the value is smaller than the current conference's running time an error is displayed. In this case the current conference will not be affected.
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. A value of 0 means that the limit specified on the conference template is used.

factoryOverridePIN	String	<p>If present, this is the String of numeric digits that participants with host privileges need to enter to join the conference. It overrides the PIN defined on the conference template. For a meeting-type conference this is the PIN used for meeting participants.</p> <p>Only new participants joining the conference are challenged for the new PIN. Existing participants are only re-challenged if the call drops.</p> <p>The String must be less than 32 characters.</p>
factoryOverrideGuestPIN	String	<p>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.</p> <p>Only new participants joining the conference are challenged for the new PIN. Existing participants are only re-challenged if the call drops.</p> <p>The String must be less than 32 characters.</p>
factoryResourceLimits	resourcelimits struct	<p>The resource limits for this conference, made up of signalling, media and licences for a conference bridge of type <i>tsmcu</i> and made up of ports for a conference bridge of type <i>mcu</i>.</p> <p>The values for this struct must not be smaller than the actual resource usage of the current conference.</p>
→ signalling	Integer	<p>Limit on the number of participants/calls that this conference may use. This is only applicable to conference bridges of type <i>tsmcu</i>.</p> <p>Accepted values are 0 to 2400.</p> <p>Use 0 to create a conference with an unlimited number of participants/calls. The number of participants will be limited by the capacity of the conference bridges in this case.</p>
→ media	Integer	<p>Limit on the media resources that this conference may use. This is only applicable to conference bridges of type <i>tsmcu</i>.</p>
→ licences	Integer	<p>Limit on the licence resources that this conference may use. This is only applicable to conference bridges of type <i>tsmcu</i>.</p>
→ ports	Integer	<p>Limit on the number of ports that this conference may use. This is only applicable to conference bridges of type <i>mcu</i>.</p> <p>Accepted values are 0 to 2400.</p> <p>Use 0 to create a conference with an unlimited number of participants/ports. The number of participants will be limited by the capacity of the conference bridges in this case.</p>

factoryLayout	String	<p>The layout that the conference participants will see when they join the conference. The parameter allows you to override the layout specified within the layout parameter of the ConfBundle object in the Provisioning API.</p> <p>If factoryLayout is omitted, the layout specified within the ConfBundle object is not overridden.</p> <p>The accepted values for factoryLayout are:</p> <ul style="list-style-type: none"> ■ <i>equal</i>: conference participants are shown in a grid pattern of equal sized panes, up to 4x4. (Not applicable to multiscreen endpoints) ■ <i>active</i>: 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. ■ <i>prominent</i>: 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) ■ <i>single</i>: the active speaker is shown in one full-screen pane. <p>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.</p> <p>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.</p>
locked	Boolean	<p>Whether the conference is locked or not.</p> <p>If a conference is locked, it continues to run with its existing participants, but no new participants can dial in. It is still possible for the API client to add more participants to the conference using the call participant.add.</p>

Returned data

Parameters	Type	Description
status	String	On success: "operation successful" On failure: "error"
info	String	On success: message saying that the conference has been modified On failure: error message
faultCode	String	This is returned in some failure cases.

factory.health.query

This call returns system status information.

Note: We recommend that clients avoid calling **factory.health.query** more frequently than once every 5 minutes to avoid performance implications.

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.

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 6GB
- on VM systems running on BE6K (Business Edition 6000) and BE7K (Business Edition 7000) RAM is 4GB
- on appliance systems (systems that report anything other than N/A in the **fan_1** to **fan_3** values) RAM is 4GB

factory.webex.add

This call adds a WebEx conference to an existing conference on the TelePresence Conductor.

Input parameters

Required parameters

Parameters	Type	Description
conferenceName	String	The name of the conference on TelePresence Conductor to which this WebEx conference should be added.
sipUri	String	The SIP URI for the WebEx conference. When using the SIP method to add 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 add a WebEx conference.
bestEffort	Boolean	<p>Whether the TelePresence Conductor will try to add this WebEx conference to the primary conference bridge or (if there is cascading) to a cascade conference bridge, even if insufficient resources have been reserved. Insufficient resources will have been reserved if the factory.conferencecreate parameter webEx was set to <i>None</i> or to the incorrect type of WebEx conference.</p> <p>This parameter is not relevant when the settings for the factory.conferencecreate parameter webEx and the settings for factory.webex.add match.</p> <p><i>True</i> will result in the TelePresence Conductor attempting to add this WebEx conference with the supplied settings to the appropriate conference bridge hosting this conference. The TelePresence Conductor will not succeed if the primary conference bridge and any associated cascade conference bridges are fully utilized.</p> <p><i>False</i> will result in this factory.webex.add call failing, if the settings specified in the call do not match the settings of the factory.conferencecreate parameter webEx.</p>

Returned data

Parameters	Type	Description
info	String	On success: "WebEx successfully added to conference "<conference name>""
status	String	On success: "operation successful".

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 \[p.22\]](#) 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 `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>idl</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>
```

```

        </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 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 specified in **receiverURI**.

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.
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.

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
<code>receivers</code>	Array	The Array of receivers containing the following:
→ <code>receiverURI</code>	String	Fully-qualified URI identifying protocol and remote device, for example <code>http://10.2.134.40:5050/RPC2</code>
→ <code>sourceIdentifier</code>	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.
→ <code>index</code>	Integer	The 'slot' for the receiver.

feedbackReceiver.extended.remove

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

Input parameters

Required parameters

Parameters	Type	Description
<code>receiverIndex</code>	Integer	The index returned by a <code>feedbackReceiver.extended.configure</code> 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	<p>The number of screens to use on a participant's endpoint.</p> <p>This parameter is only relevant if the conference template associated with the conference that this participant has dialed into:</p> <ul style="list-style-type: none"> 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. <p>If the conditions above are all met, the number of screens specified on the conference template is overridden with the number specified in factoryNumScreens.</p> <p>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'.</p>
addAsGuest	Boolean	<p>Whether the participant is added as a guest on the conference.</p> <p>If the conference is of type Meeting, the parameter defaults to <i>False</i> and is rejected when changed to <i>True</i>.</p> <p>If the conference is of type Lecture, the parameter defaults to <i>True</i> and must be changed to <i>False</i> if the participant should be added as a host.</p>
participantProtocol	String	The protocol to use for outdial calls: either <i>sip</i> or <i>h323</i> . In a deployment using the TelePresence Conductor's B2BUA this setting must be <i>sip</i> .
participantType	String	The TelePresence Conductor ignores any values specified for this parameter and always enforces the value to be <i>ad_hoc</i> .

For conferences hosted on a TelePresence MCU all other optional parameters are documented in [Cisco TelePresence MCU API reference guide](#).

Returned data

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

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: <i>h323</i> , <i>sip</i> or <i>vnc</i> .
participantType	String	The type of the participant: <i>by_address</i> , <i>by_name</i> , or <i>ad_hoc</i> .

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.34].

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: <i>h323</i> , <i>sip</i> or <i>vnc</i> .
participantType	String	The type of the participant: <i>by_address</i> , <i>by_name</i> , or <i>ad_hoc</i> .

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
<code>factoryConferenceIds</code>	Array	An Array of <code>factoryConferenceId</code> 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
<code>participants</code>	Array	An Array of participant structs as returned by the conference bridge API.
→ <code>mcuIpAddress</code>	String	IP address of the conference device holding the participant. Added by TelePresence Conductor.
→ <code>factoryConferenceId</code>	String	Conference identifier supplied by the TelePresence Conductor for tracking. It identifies the conference the participant belongs to.
→ <code>participantName</code>	String	The unique name of a participant. For TelePresence Servers this is a unique participantID. For TelePresence MCUs this is a unique participantName.
→ <code>participantType</code>	String	One of <i>by_address</i> , <i>by_name</i> or <i>ad_hoc</i> .
→ <code>conferenceName</code>	String	The name of the conference.

Parameters	Type	Description
→ 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 <i>tsmcu</i> or <i>mcu</i> .
→ audioRxMuted	Boolean	<i>true</i> means that audio from this participant will not be heard by other conference participants.
→ audioTxMuted	Boolean	<i>true</i> means that the conference bridge does not send the audio part of the conference to this participant.
→ videoTxMuted	Boolean	<i>true</i> 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	<p><i>None</i> represents a conference that does not support WebEx. (This is the default parameter.)</p> <p><i>SIP</i> represents a conference that supports a SIP WebEx conference, where resources for one call is used on the primary conference bridge.</p> <p><i>SIP-TSP</i> 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.</p>
→ factoryWebExCallType	String	One of <i>VideoAudio</i> , <i>Video</i> or <i>Audio</i> .

Parameters	Type	Description
→ factoryCallState	String	<p>The current state of the participant as reported by the conference bridge hosting the conference.</p> <p>The corresponding conference bridge state is translated to one of these states:</p> <ul style="list-style-type: none"> ■ <i>disconnected</i> ■ <i>ringing</i> ■ <i>connected</i> ■ <i>awaitingTrigger</i> ■ <i>callLegFailed</i> ■ <i>retrying</i>.

Additional parameters contained in a **participant** struct, as documented in [Cisco TelePresence MCU API reference guide](#).

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	<p>The unique name of a participant.</p> <p>For TelePresence Servers this is a unique participantID.</p> <p>For TelePresence MCUs this is a unique participantName.</p>
message	String	<p>The message String to send to the participant.</p> <p>For TelePresence Servers the maximum number of allowed characters is 500.</p> <p>For TelePresence MCUs the maximum number of allowed characters is 255.</p>

Optional parameters

Parameters	Type	Description
participantProtocol	String	The protocol: <i>h323</i> , <i>sip</i> or <i>vnc</i> .
participantType	String	The type of the participant: <i>by_address</i> , <i>by_name</i> , or <i>ad_hoc</i> .

factoryPosition	String	<p>The position of the message on the participant's screen.</p> <p>The options are:</p> <ul style="list-style-type: none"> ■ top-left ■ top-centre ■ top-right ■ middle-left ■ middle-centre ■ middle-right ■ bottom-left ■ bottom-centre ■ bottom-right <p>The positions map to approximate positions on the relevant conference bridge type.</p> <p>The default is middle-centre.</p>
duration	Integer	<p>The length of time, in seconds, that the message is displayed for on the participant's screen.</p> <p>The value must be greater than or equal to 0.</p> <p>The default is 30.</p>
durationSeconds	Integer	<p>The length of time, in seconds, that the message is displayed for on the participant's screen.</p> <p>This parameter is deprecated. Use duration instead.</p>

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: <i>h323</i> , <i>sip</i> or <i>vnc</i> .
participantType	String	The type of the participant: <i>by_address</i> , <i>by_name</i> , or <i>ad_hoc</i> .

Optional parameters

Parameters	Type	Description
------------	------	-------------

displayNameOverrideStatus	Boolean	<i>true</i> 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 <i>true</i> .
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	<i>true</i> means that audio from this participant will not be heard by other conference participants.
audioRxGainMode	String	<i>none</i> , <i>automatic</i> , or <i>fixed</i>
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	<i>true</i> means that video from this participant will not be seen by other conference participants.
videoTxWidescreen	Boolean	If <i>true</i> , the TelePresence MCU sends video in a form suitable for a widescreen 16:9 display to this participant.
autoDisconnect	Boolean	<i>true</i> 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. <i>false</i> means this endpoint requires manual disconnection.
suppressDtmf	String	Controls the muting of DTMF tones. One of <i>fecc</i> , <i>always</i> , <i>never</i> , or <i>default</i> .
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	<i>true</i> 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. Value must be in the range from 0 to 3. 0 means that there is no border.
important	Boolean	<i>true</i> means this participant's video is important; it will dominate the layout.

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 \[p.28\]](#), which was deprecated in version XC2.3.

Input parameters

Required parameters

Parameters	Type	Description
<code>conferenceName</code>	String	The name of the conference.
<code>participantName</code>	String	The unique name of a participant.
<code>participantProtocol</code>	String	The protocol: <i>h323</i> , <i>sip</i> or <i>vnc</i> .
<code>participantType</code>	String	The type of the participant: <i>by_address</i> , <i>by_name</i> , or <i>ad_hoc</i> .

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.
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 - The device has reached the limit of the number of participants that can be configured and no more can be created.

Fault Code	Description
8	no conference name or auto attendant id supplied - A conference name or auto attendant identifier is required, but is not present.
9	no participant name supplied - A participant name is required but is not present.
10	no participant address supplied - A participant address is required but is 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 is 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 is used which does not correspond to any participant type known to the device.
21	no conference alias supplied - A conference alias is required but is 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 was not 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.
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 cannot 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 .

Fault Code	Description
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 \[p.37\]](#)
- [Provisioning API \[p.43\]](#)
- [SIP Domain API \[p.60\]](#)

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 \[p.63\]](#).

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, in order to support creating scheduled conferences.

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/4.0/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 \[p.41\]](#)
- [ParticipantCost object \[p.39\]](#)
- [Resources object \[p.38\]](#)

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 <i>ports</i> . For a TelePresence Server the dimension can be one of <i>signalling</i> , <i>media</i> or <i>licences</i> . 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 <i>mcu</i> for a TelePresence MCU or <i>tsmcu</i> for a TelePresence Server.
bridge_capabilities	Array [String]	An Array of capabilities that are supported by the conference bridge. The Strings <i>cascading</i> , <i>multiscreen</i> , <i>role_by_pin</i> and <i>centralized_licensing</i> 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	<p>The value for Maximum screens that has been defined on the associated conference template. This can be in the range of 1 to 4.</p> <p>For TIP-compliant endpoints dialing into a rendezvous conference using the TelePresence Conductor's B2BUA the value is the maximum limit of supported screens.</p> <p>For reserved hosts, policy service routed calls and ad hoc escalated calls the value is the default number of screens for initial resource calculations.</p> <p>The value is always 1 for conference bridges that do not support multiscreen endpoints (where bridge_capabilities does not include <i>multiscreen</i>).</p>
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	<p>The value defined for Maximum when limiting the number of participants on the associated conference template.</p> <p>0 means that there is no limit set.</p> <p>The value includes any auto-dialed participants and reserved hosts defined for the conference template.</p>
pool_capacity	Array [Resources]	<p>A JSON object containing the resource capacity, which is potentially available across all the pools marked to be used for scheduling within a specific Service Preference. It does not take into account the resource usage for conferences that are currently running or the resources available in fallback pools that are included in the Service Preference, but not marked to be used for scheduling.</p> <p>Pools can be marked to be used for scheduling on the Service Preference page within the TelePresence Conductor user interface.</p>
conference_cost	Array [Resources]	A JSON object indicating any one-off per-conference costs.
participant_cost	Array [ParticipantCost]	<p>An Array of JSON objects containing the total costs for the participant taking into consideration multiple screens. If bridge_capabilities includes <i>multiscreen</i>, there will be 4 elements in the Array. If bridge_capabilities does not include <i>multiscreen</i>, there will only be 1 element in the Array.</p> <p>The client can use the appropriate Array element for the actual number of screens to work out the required costs.</p>

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/4.0/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 \[p.44\]](#)
- [ServiceInfo object \[p.46\]](#)
- [ConfBundleID object \[p.47\]](#)
- [ConfBundleVer object \[p.48\]](#)
- [ConfBundle object \[p.48\]](#)
- [Alias object \[p.55\]](#)
- [AutoDialParticipant object \[p.56\]](#)

The **QualityInfo** data structure is a JSON object containing an Array of **QualitySpec** objects. The **QualitySpec** objects contain the quality settings that have been defined on this TelePresence Conductor.

The **ServiceInfo** data structure is a JSON object containing an Array of **ServiceRecord** objects. The **ServiceRecord** objects contain all Service Preferences defined on this TelePresence Conductor. A **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 **AutoDialParticipant** 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 **AutoDialParticipant** 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.

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

QualityInfo object

The **QualityInfo** data structure is a JSON object containing an Array of **QualitySpec** objects. The **QualitySpec** objects contain the 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 .
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 <i>content</i> or <i>video</i> .
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 object containing an Array of **ServiceRecord** objects. The **ServiceRecord** objects contain all Service Preferences defined on this TelePresence Conductor. A **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.

Attribute name	Type	Description
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.
bridge_capabilities	Array [String]	An Array of capabilities that are supported by the conference bridge. The Strings <i>cascading</i> , <i>multiscreen</i> , <i>role_by_pin</i> and <i>centralized_licensing</i> 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. If <i>centralized_licensing</i> is included in the Array, TelePresence Conductor informs the API client that it can provide owner information via its ConfBundle PUT operations.

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": ["cascading", "multiscreen", "centralized_licensing"]
  }
]
```

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 **AutoDialParticipant** 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

Attribute name	Type	Description	Mandatory or 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, <i>false</i> is implied.	optional
content_quality	String	<p>The UUID (a globally unique ID) pointing to the content quality to be used for this conference.</p> <p>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]).</p> <p>The quality_type attribute of the QualitySpec data structure must have the value <i>content</i>.</p> <p>This attribute is ignored if allow_content is <i>false</i>.</p>	mandatory (if allow_content is <i>true</i>)
host_quality	String	<p>The UUID (a globally unique ID) pointing to the host quality to be used for this conference.</p> <p>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]).</p> <p>The quality_type attribute of the QualitySpec data structure must have the value <i>video</i>.</p> <p>If the associated aliases for this ConfBundle have a role of <i>by_pin</i>, host_quality must be the same as guest_quality.</p>	mandatory
guest_quality	String	<p>The UUID (a globally unique ID) pointing to the guest quality to be used for this conference.</p> <p>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]).</p> <p>The quality_type attribute of the QualitySpec data structure must have the value <i>video</i>.</p> <p>If the associated aliases for this ConfBundle have a role of <i>by_pin</i>, host_quality must be the same as guest_quality.</p>	mandatory

Attribute name	Type	Description	Mandatory or optional
layout	String	<p>An enum String that defines the video layout scheme to be seen by participants joining conferences created from this ConfBundle.</p> <p>The layout must be one of the following types, which have been defined in this API:</p> <ul style="list-style-type: none"> ■ <i>equal</i>: conference participants are shown in a grid pattern of equal sized panes, up to 4x4. (Not applicable to multiscreen endpoints) ■ <i>active</i>: 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. ■ <i>prominent</i>: 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) ■ <i>single</i>: the active speaker is shown in one full-screen pane. <p>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.</p> <p>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.</p> <p>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.</p>	optional
advanced_parameters	Object	<p>A conference bridge specific JSON object for configuring advanced conference parameters on conference creation.</p> <p>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.53] for more information about prohibited and discouraged parameters.</p> <p>We strongly recommend that cascade_advanced_parameters are supplied with the same JSON object as advanced_parameters for the primary conference bridge. The attribute cascade_advanced_parameters may be removed in a future release of the TelePresence Conductor.</p>	optional

Attribute name	Type	Description	Mandatory or 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 <i>multiscreen</i> . If allow_multiscreen is omitted, the value is implied to be <i>false</i> .	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 <i>false</i> . 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 <i>mcu</i> . If optimize_resources is omitted, it is implied to be <i>true</i> .	optional
reserved_hosts	Integer	The number of hosts for whom resources should be reserved. If reserved_hosts is omitted, it is implied to be '1'. If the associated aliases for this ConfBundle have a role of <i>by_pin</i> , reserved_hosts is ignored. This is because the TelePresence Conductor cannot differentiate between hosts and guests dialing into the conference.	optional
reserved_cascades	Integer	The maximum number of cascades this conference is allowed to have. This attribute is ignored if the bridge_capabilities attribute within the ServiceInfo data structure does not include <i>cascading</i> . If reserved_cascades is omitted, it is implied to be '0'.	optional

Attribute name	Type	Description	Mandatory or optional
cascade_advanced_parameters	Object	<p>A conference bridge specific JSON object for configuring advanced conference parameters for use on cascade conference bridges.</p> <p>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.53] for more information about prohibited and discouraged parameters.</p> <p>We strongly recommend that cascade_advanced_parameters are supplied with the same JSON object as advanced_parameters for the primary conference bridge. The attribute cascade_advanced_parameters may be removed in a future release of the TelePresence Conductor.</p> <p>The cascade advanced parameters must not include the parameters pin and/or guestPin (for TelePresence MCUs). 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.</p> <p>This attribute is ignored if the bridge_capabilities attribute within the ServiceInfo data structure does not include <i>cascading</i>.</p>	optional
scheduled	Boolean	<p>Whether or not conferences generated from this ConfBundle are scheduled. If the value is <i>true</i> the conference can only be created via the API call factory.conferencecreate and not by participants dialing the conference alias.</p> <p>If scheduled is omitted, it is implied to be <i>false</i>.</p>	optional
guests_wait_for_host	Boolean	<p>Whether or not guests should wait for a host to join a conference before they can join.</p> <p>This setting is only applicable to TelePresence Server hosted conferences. It is ignored for TelePresence MCUs, where guests must always wait for a host to join.</p> <p>The default is <i>false</i>.</p>	optional
auto_dialed_participants	Array [AutoDialedParticipant]	<p>An Array of AutoDialedParticipant objects, with 1 to 10 elements.</p> <p>If auto_dialed_participants is omitted, it is implied that there are no auto-dialed participants in this conference.</p>	optional
host_pin	String	<p>The PIN required for hosts to join the conference.</p> <p>If host_pin is omitted, it is implied that there is no PIN required.</p> <p>If the associated aliases for this ConfBundle have a <i>role</i> of <i>by_pin</i>, host_pin must be provided.</p>	optional

Attribute name	Type	Description	Mandatory or 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
owner_id	String	The GUID that links this ConfBundle object to its matching Owner object. This GUID points to the same user as external_owner_id .	optional
external_owner_id	String	The GUID that links to the Active Directory entry for the conference initiator. The TelePresence Conductor tries to match the conference initiator ID supplied via a conference.create or factory.conferencecreate call to this String. This GUID points to the same user as owner_id .	optional
multi_party_license	String	The type of multiparty license associated with this ConfBundle . <i>PMP</i> is a personal multiparty license. <i>SMP</i> is a shared multiparty license. If omitted, the default is <i>SMP</i> .	optional

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
<code>role</code>	String	<p>An enum String describing the role for this conference alias; either <i>host</i>, <i>guest</i> or <i>by_pin</i>.</p> <p>The option <i>by_pin</i> may only be chosen if:</p> <ul style="list-style-type: none">■ The <code>service_preference_id</code> for the corresponding ConfBundle points to a ServiceInfo object that includes <i>by_pin</i> in its <code>bridge_capabilities</code>. This means that the conference bridge type that will host this conference supports determining role by PIN.■ All other Alias objects for the corresponding ConfBundle have a <code>role</code> of <i>by_pin</i>. <p>For more information on setting the option <i>by_pin</i> see Role determined by PIN [p.57].</p>	mandatory

Attribute name	Type	Description	Mandatory or optional
exact_match	String	<p>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.</p> <p>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.</p> <p>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.60] to replace the IP address with the SIP domain.</p>	mandatory
allow_conference_creation	Boolean	<p>Whether participants dialing this conference alias can create the conference or not. If the value is <i>false</i> 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 <i>true</i>.</p> <p>If allow_conference_creation is omitted, it is implied to be <i>true</i>.</p>	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	<p>An enum String describing the role for this auto-dialed participant; either <i>host</i> or <i>guest</i>.</p> <p>If the aliases for the ConfBundle associated with this AutoDialedParticipant have a role of <i>by_pin</i>, this auto-dialed participant's role must be set to <i>host</i>.</p>	mandatory
participant_quality	String	<p>The UUID (a globally unique ID) pointing to the maximum quality to be used for this auto-dialed participant.</p> <p>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]).</p> <p>The quality_type attribute of the QualitySpec data structure must have the value <i>video</i>.</p>	mandatory

Attribute name	Type	Description	Mandatory or optional
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 <i>true</i> . If keep_conference_alive is omitted, it is implied to be <i>false</i> . If the aliases for the ConfBundle associated with this AutoDialedParticipant have a role of <i>by_pin</i> , this auto-dialed participant's keep_conference_alive parameter must be set to <i>false</i> or omitted.	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 <i>False</i> , the auto-dialed participant will not be dialed when the conference is created. If enabled_state is omitted, it is implied to be <i>True</i> .	optional

The **participant_protocol** attribute cannot be changed, it is always set to SIP.

Role determined by PIN

The role for an alias can either be specified explicitly, as *host* or *guest*, or the role can be determined by PIN. For more information on host and guest roles see About host and guest roles in [Cisco TelePresence Conductor Administrator Guide](#) or in the Online Help.

If the role is supposed to be determined by PIN, all **Alias** objects for a specific **ConfBundle** must have a role of *by_pin*. Instead of TelePresence Conductor determining the privileges that a participant gets after dialing an alias with a particular role, the conference bridge determines the privileges that the participant gets, based on the PIN that was entered.

Prerequisites

- TelePresence Conductor API for version XC3.0 or later
- TelePresence Server version 4.1 or later
- Cisco TMSPE version 1.4 or later

Limitations

- Only conferences hosted on TelePresence Servers are supported. Conferences on TelePresence MCUs are not supported.
- Only conferences provisioned via the TelePresence Conductor Provisioning API are supported. Conferences configured via the TelePresence Conductor user interface are not supported.

Configuration requirements

- Within a **ConfBundle**, either all aliases must have a **role** of *by_pin*, or none of them may have a **role** of *by_pin*.
- The parameter **host_pin** must be provided. The parameter **guest_pin** may be provided. If **guest_pin** is omitted, it is implied that there is no guest PIN.

- Both **host_quality** and **guest_quality** must be provided and they must have the same value. The TelePresence Conductor is unable to differentiate between hosts and guests. Only once the participant has entered the PIN can the conference bridge itself differentiate between hosts and guests.
- All auto-dialed participants that are associated with this **ConfBundle** must have the **role** set to *host*.
- All auto-dialed participants that are associated with this **ConfBundle** must have **keep_conference_alive** set to *false* or omitted (which implies false).
- If **reserved_hosts** is set on the **ConfBundle**, its value is ignored. The TelePresence Conductor is unable to determine whether a participant dialing in is a host or a guest. It could therefore happen that the maximum number of participants allowed in the conference (specified in **max_participants**) is filled up purely with guests, not allowing any hosts to dial into the conference.

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/4.0/serviceinfo/
```

Retrieving an individual ServiceRecord object

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

```
/api/4.0/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/4.0/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/4.0/qualities/<QualityInfo.quality_group_name>
```

ConfBundles operation

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 **ConfBundleIDs** that point to the corresponding **ConfBundleVer**) the client must perform the following operation:

GET **/api/4.0/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-244862180000": "4a2b393a-910d-435f-946b-31e34813bbbb",
  "4b54f5ff-2bc2-4823-8173-fdd460b51111": "d0e475eb-c657-4849-a436-935633ab5555",
  "c218e5eb-589a-4d1c-84f6-46e387422222": "606297c5-4d60-44c8-962c-1a9a6fba4444"
}
```

Retrieving a specific **ConfBundle** object

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

GET **/api/4.0/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/4.0/confbundles/**

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/4.0/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/4.0/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/4.0/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
<code>SIPDomain</code>	String or null	<p>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.</p> <p>The <code>SIPDomain</code> can be either:</p> <ul style="list-style-type: none">■ 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/4.0/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/4.0/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.

Owners API

The **Owners** API has an **Owners** object defined that contains an array of **Owner** objects.

The **Owner** object is a subset of the **ConfBundle** object. It contains information about the initiator of a conference, including his or her multiparty license. **ConfBundle** objects take precedence over **Owner** objects.

One of the clients using this API is Cisco TMSPE. TelePresence Conductor creates a new **Owner** object whenever Cisco TMSPE informs it about a new CMR that has been created. TelePresence Conductor creates a **ConfBundle** object whenever Cisco TMSPE informs it about a new CMR being claimed. CMRs are claimed when a user personalizes (for example creates a PIN) and activates it from the User Portal.

Owners object

The **Owners** data structure is a JSON object containing the **Owner** object. It allows for bulk provisioning.

Attributes

Attribute name	Type	Description
owners	Array [Owner]	An Array of Owner objects. The maximum number of elements is 100.

Attribute name	Type	Description
generation	String	A UUID that identifies the generation of the owners Array. It shows which active directory sync resulted in this Array and can be used to delete old records.

JSON example

The following is an example of an **Owners** JSON object:

```
{
  "owners": [
    {
      "owner_id": "aabbccdd-1234-5678-abcd-aaabbbcccd",
      "external_owner_id": "12345678-90ab-cdef-4321-012345678912",
      "multi_party_license": "PMP"
    },
    {
      "owner_id": "a1b2c3d4-abcd-dcba-efgh-aa11bb22cc33",
      "external_owner_id": "abcdefgh-1111-2222-3333-112233445566",
      "multi_party_license": "SMP"
    }
  ],
  "generation": "98765432-aaaa-bbbb-cccc-aaabbbcccd"
}
```

Owner object

The **Owner** data structure is a JSON object containing a subset of the information that is contained in a **ConfBundle**.

Attributes

Attribute name	Type	Description
owner_id	String	The GUID that links this Owner object to its matching ConfBundle object. This GUID points to the same user as external_owner_id .
external_owner_id	String	The GUID that links to the Active Directory entry for the conference initiator. The TelePresence Conductor tries to match the conference initiator ID supplied via a conference.create or factory.conferencecreate call to this String. This GUID points to the same user as owner_id .
multi_party_license	String	The type of multiparty license associated with this ConfBundle . <i>PMP</i> is a personal multiparty license. <i>SMP</i> is a shared multiparty license. If omitted, the default is <i>SMP</i> .

Owners operation

owners is a REST operation. It includes a PUT and two DELETE methods. It does not include any POST or GET methods.

Updating one or more Owner objects

Perform the following operation and include a list of the **Owner** objects to update:

PUT /api/4.0/owners

Deleting all Owner objects that are not associated with a provisioned ConfBundle

Perform the following operation:

DELETE /api/4.0/owners

Deleting all Owner objects that do not belong to the specified generation and are not associated with a provisioned ConfBundle

Perform the following operation:

DELETE /api/4.0/<Owners.generation>

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. The information provided may vary.
- **details**: Informal information about the error to aid debugging. 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'conference
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.
- **https://<IP address of the Conductor>/api/4.0/multipartylicense** - summary information about the number of licenses that have been added as an option key to the TelePresence Conductor. The totals apply across all peers of a cluster.

External REST API

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/mcstatus**
- **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**)

Multipartylicense REST API

The TelePresence Conductor supports a GET method for the REST API **https://<IP address of the Conductor>/api/440.10/multipartylicense**.

The GET method returns the total number of licenses for each type of multiparty license. The numbers represent only licenses that have been added to the TelePresence Conductor as option keys, not licenses actually used. The totals apply across all peers of a cluster.

The JSON returned has the following format:

```
{pmp_a_total: <int>
pmp_b_total: <int>
smp_total: <int>}
```

System information

The version of the TelePresence Conductor software that is running, can be obtained using **https://<IP address of the Conductor>/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.

Scheduling via Cisco TMS

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

- [Capacity Management API \[p.37\]](#)
The main purpose of the Capacity Management API is to provide the API client with the capacity information for a conference, in order to support creating scheduled conferences.
- [factoryResourceLimits \[p.14\]](#) struct within `factory.conferencecreate`
The parameter `factoryResourceLimits` limits the resources for a conference and helps to prevent arbitrary conference enlargement.

The TelePresence Conductor user interface provides the following features:

- You can create conference bridge pools on the TelePresence Conductor. To each pool you can add one or more conference bridges.
- You can then add one or more pools to Service Preferences.
- Within a specific Service Preference, pools can be marked to be used for scheduling. This can be done on the **Service Preference** page of the TelePresence Conductor user interface. The capacity information returned to Cisco TMS represents the sum of the capacities of all the conference bridges in the marked pool(s).

The following restrictions apply to scheduling via Cisco TMS:

- 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. Dedicated bridge scheduling is only supported where a Service Preference contains a single conference bridge. If there is more than one conference bridge in a pool and cascading is enabled, resources are reserved for the cascade link. These resources are used whether the cascade is created or not.
- Pre-configured endpoints defined on the TelePresence Conductor are not considered in the capacity calculations and are therefore not supported in dedicated bridge scheduling.
- Dedicated content ports on the TelePresence MCU are not considered in the capacity calculations.
- For dedicated bridge scheduling, the number of hosts to reserve should be set to 1. The Capacity Management API will assume this value. Reserved resources will not be included in the `ServiceParams` data structure.
- The Cisco TelePresence T3 multiscreen endpoint is not supported and scheduling may not handle the T3 correctly.

For information on the scheduling scenarios that can be configured on the TelePresence Conductor, see [Cisco TelePresence Conductor Admin Guide](#) or the online help.

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 that supports 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}

The per conference cost is 1 because the conference supports content.

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

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 that does not support 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}

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}

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>`

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.

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

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

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

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

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

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

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

© 2015 Cisco Systems, Inc. All rights reserved.