Cisco TelePresence Management Suite Extension Booking API

Programming Reference Guide

Last Updated: April 2018

API version 18

(Cisco TMS 15.4)

Introduction

Cisco TelePresence Management Suite Extension Booking API (Cisco TMSBA) gives developers access to Cisco TelePresence Management Suite (Cisco TMS) booking functionality. The API is also used by the Cisco TMS extensions for Microsoft Exchange and IBM Lotus Notes, and the Cisco TMSPE Smart Scheduler.

This document details the objects and entities used by Cisco TMSBA, includes references of the functions and objects available in the booking and remote setup APIs, explains the versioning logic,and provides usage patterns suggesting how an external user interface, booking database, or booking system may interact with Cisco TMS using the APIs.

The target audience for this document is developers seeking to implement a data/audio/video conferencing booking solution that is not supported by Cisco TMS directly, or where existing Cisco TMS features do not provide the necessary interoperability. Such booking systems will be referred to as external booking systems in this document.

Changes between API versions

Changes to Cisco TMSBA are not tracked in this document, but can be found in the release notes for the corresponding version of Cisco TMS. This includes unversioned changes that will apply to every version of Cisco TMSBA called with that version of Cisco TMS.

Related documents

The following table lists documents and websites referenced in this document, and other supporting documentation. All documentation for the latest version of Cisco TelePresence Management Suite Booking API can be found at: http://www.cisco.com/en/US/products/ps11338/tsd_products_support_series_home.html

Title	Link
Cisco TelePresence Management Suite Release Notes	http://cisco.com
SOAP Version 1.2	http://www.w3.org/TR/soap12/
Web Services Description Language (WSDL) 1.1	http://www.w3.org/TR/wsdl
WSDL 1.1 Binding Extension for SOAP 1.2	http://www.w3.org/Submission/wsdl11soap12

API Overview

This chapter provides an overview of the basic principles of Cisco TMSBA.

Functional Overview 4
Cisco TMS Entities 5
API Versioning 6

Functional Overview

Cisco TMSBA makes it possible to let users book resources from custom booking applications (in this document referred to as "clients"), integrating the custom booking application with the Cisco TMS reservation database.

Using Cisco TMSBA to interact with Cisco TMS is the only safe and supported way of integrating third party or custom booking applications with Cisco TMS. The Cisco TMSBA is a versioned and stable interface into Cisco TMS, where backward compatibility is guaranteed in new releases.

Main Features

Cisco TMSBA has three main features:

- Importing and/or displaying Cisco TMS-managed resources in a client or booking system external to Cisco TMS.
- Reading and displaying system availability information and bookings from the reservation database of Cisco TMS. This information can be used for replicating bookings from Cisco TMS to the external booking system.
- Forwarding booking requests made by an external booking system to Cisco TMS. Forwarding a booking request to Cisco TMS lets the client utilize the routing logic of Cisco TMS, which determines whether network resources like an Multipoint Control Unit (MCU) are required, and automatically reserves these resources. The client therefore does not need to worry about infrastructure resources such as gateways and MCU port availability.

Booking Ownership

The client can choose whether to authenticate with Cisco TMS as a service account or as the user actually making the booking. If authenticating as a service account, the client must explicitly specify the actual owner of the booking when performing the `SaveConferenceWithMode` function. If choosing this option, it is important that the service account is a member of a group having *Book on behalf of* permissions in Cisco TMS.

`IsTMSBookOnBehalfUser` and `IsBookonBehalfOfUser` in the remote setup API can be used to assess whether a service account or other user has *Book on behalf of* permissions. The `GenerateConferenceAPIUser` function can be used for creating a service account. See Remote Setup API Reference, page 18.

Conference Routing

When booking through the Cisco TMSBA, clients book endpoints only. Network infrastructure products, such as gateways and Multipoint Control Units (MCUs) are automatically added by Cisco TMS if needed. The settings under **Administrative Tools > Conference Settings > Advanced Conference Options** in Cisco TMS are used when routing conferences created through Cisco TMSBA.

Replication

Cisco TMSBA supports two-way replication between the client and Cisco TMS. If the client maintains its own copy of the reservation database, it must periodically ask Cisco TMS for changes.

In Cisco TMS, all changes to bookings are tracked in the Cisco TMS (tmsng) database. The data in this table is available in Cisco TMSBA through the `GetTransactionsSince` function. This function gives the client a list of recent changes on the Cisco TMS side. This function should be run fairly often (such as every five minutes) so that the client and Cisco TMS reservation databases are in a consistent state.

Booking Modes

The client can request one of two booking modes from Cisco TMS:

- *Strict*–all conferences that cannot be scheduled exactly as requested will cause Cisco TMS to throw an exception, and the conferences will not be saved.
- *BestEffortForced*–conferences that have a resource conflict or routing issue will be saved in Cisco TMS as *Defective*, which is shown to the client as a conferencestate on the returned conference object. This mode works as described below.

Defective Conferences

A *Defective* conference in Cisco TMS has been booked by an external client that encountered a resource conflict or routing problem.

A defective conference retains all properties of the booking request without setting up routing or consuming telepresence resources. Until all issues are resolved, Cisco TMS will not initiate a defective conference or send it to endpoints.

- In the case of a routing issue, all endpoints in the booking will be set to *Busy* for the scheduled time, keeping the reservation while the administrator or user resolves the issue.
- In the rare case of an endpoint reservation conflict, the endpoints will not be set to *Busy* for the defective booking.

Defective conferences can be corrected by the organizer or the administrator:

- Users who book conferences that are saved as defective will be notified by email and can resolve most issues by changing their request and rescheduling from their client.
- Administrators can locate and resolve defective conferences in Cisco TMS by going to **Administrative Tools > Diagnostics > Conference Diagnostics** or **Booking > List Conferences**.

  Conferences that are defective because of configuration errors or a permanent lack of routing resources must be resolved by an administrator.

When scheduling a series where only some occurrences have a resource conflict or routing issue, Cisco TMS will only store the problematic occurrences as defective, leaving the remaining occurrences unaffected.

If you apply the *BestEffortForced* booking mode, Cisco TMS will apply the same logic to bookings from your client, and return *defective* on conferences that have a resource conflict or routing problem.

Cisco TMS Entities

System

A system entity in Cisco TMS is any item that can be booked. Note that systems provisioned using Cisco TelePresence Management Suite Provisioning Extension cannot be booked and are therefore not considered systems in this context.

There are two types of system entities:

- systems that are controlled by Cisco TMS (and thus receive phonebooks, generate CDRs and so on)
- systems added to Cisco TMS as "Unmanaged Endpoint".

Other types of entries, such as phonebook entries or provision directory users, are not system entities in Cisco TMS.

Each system entity has its own **TMS System Id**, which is used to uniquely identify the system in the Cisco TMS database, tied to system settings such as SIP URIs, gatekeeper addresses, and software versions. A subset of this system data is available in Cisco TMSBA through the `GetSystemById` function.

Conference

Each booking in Cisco TMS is a conference entity in the **tmsng** database.

Each conference has a unique combination of **ExternalSourceId** and **ExternalPrimaryKey** used to identify the conference. Each occurrence of a recurrent series is also identified by an **InstanceId**.

Using **ConferenceId** as sole identifier is considered a legacy feature, but still supported by Cisco TMSBA.

User

The Cisco TMS user entity holds information about Cisco TMS users, such as their first name, last name, username, and email address. All users have a unique user ID.

API Versioning

Cisco TMSBA versioning is designed to provide a backwards compatible API to clients. This means that applications written for an older version of the API will keep working when upgrading to Cisco TMS with a newer version of the API.

Each SOAP message sent to the API by the client must announce which version it complies with in a header. The API will filter what is returned to the client based on this announced version. See The SOAP Header, page 12.

Version History

Version 0 of the API corresponds to version 9 of Cisco TMS. Not all versions of Cisco TMS include changes to the API.

Below is an overview of the last Cisco TMS versions that have included a new version of Cisco TMSBA.

API version	Cisco TMS version
18	15.4
18	15.3
17	15.2.1
17	15.1
16	15.0
15	14.6
14	14.5
13	14.4
12	14.3
11	14.2
10	14.1.1

API version	Cisco TMS version
9	13.2
8	13.1.2
7	13.1
6	13.0.1
5	12.6

Versioning Examples

When a change to the API is needed, the WSDL must be updated. The WSDL is the contract betweeen the API and the clients and provides information covering all methods and data types, including values, that the API supports.

Time Zone Versioning

Cisco TMSBA allows each conference object to include a **ConferenceTimeZoneRules** element that contains an array of rules for the time zone (UTC offset) and daylight savings time. (For detail about this element, see ConferenceTimeZoneRules, page 35.)

This element was introduced in Cisco TMSBA version 11 with Cisco TMS 14.2. If a client is using version 10 or earlier of the API, regardless of Cisco TMS version:

- Booking dates must be in UTC.
- DayOfWeek values in recurrence patterns must be given in UTC if the API version used is 8 or later.
- Dates are provided in UTC when retrieving conference information.
- Cisco TMS uses the time zone rules of the server.

Note that these conferences will be prone to DST change errors when the time zone of the client is not the same as the configured time zone on the Cisco TMS server.

Introducing a New Bandwidth Value

When, for example, support for an additional bandwidth value is added to the API, the WSDL changes, and the version number is increased. If this new bandwidth value is added in version 9 of the API, every client declaring that they are using version 9 or later will get this new value returned from Cisco TMSBA.

Clients that announce compliance to versions earlier than 9 will *not* get the new value returned. Instead , the API returns the special value *default*. The API allows all incoming values, but filters values going out. A client declaring a version lower than 9, that then enumerates a conference with the new bandwidth value, will get the special value *default* returned.

If the client uses this conference and updates the conference with bandwidth set to default, no change to the bandwidth value is made, as the API knows this special value means no changes should be made.

This filtering of values that older clients do not understand makes the API backwards compatible; older clients built on older WSDLs can still use an API with a higher version.

Using the APIs for Remote Setup and Booking

This chapter presents the prerequisites and limitations for using the APIs for booking and remote setup, and outlines some suggestions for usage.

Usage Requirements 8
Setting up Your Environment 11
The SOAP Header 12
GUI Pattern 13
Replication Pattern 14
Limitations 15

Usage Requirements

This section describes the licensing, permission, and authentication requirements for using the APIs.

API Licensing

The API functions that require a license key are the following:

- `SaveConferenceRecInstanceWithMode`
- `SaveConferenceWithMode`
- `SaveConference`
- `SaveConferenceRecInstance`
- `SaveConferences`
- `EndConferenceById`
- `DeleteConferenceById`
- `DeleteConferenceRecInstanceById`
- `DeleteConferenceByExternalId`
- `DeleteConferenceInstanceByExternaId`
- `EndConferenceByExternalId`

The remaining API features do not require special licensing for use. Contact your Cisco reseller/partner for more licensing information.

Licensing Requirements

Each telepresence endpoint to be booked through Cisco TMSBA must already have been added to Cisco TMS and licensed for general Cisco TMS usage.

Additionally, in order to use Cisco TMSBA for booking these endpoints, you must have one of the following:

- One Cisco TMSXE – Extension for Microsoft Exchange option key per 25 telepresence endpoints integrated with Cisco TMS, usually recommended for smaller deployments. See below for detail on how system licenses are activated.
- One Cisco TelePresence Management Suite Booking API license per 25 registered endpoints per 25 telepresence endpoints integrated with Cisco TMS, usually recommended for smaller deployments. See below for detail on how system licenses are activated.
- One Cisco TelePresence Management Suite Extension for IBM Lotus Notes option key per 25 telepresence endpoints integrated with Cisco TMS, usually recommended for smaller deployments. See below for detail on how system licenses are activated.
- One Application Integration Package option key per Cisco TMSBA installation. This option is recommended for deployments with a large number of endpoints.

Note: If both license keys are present, Cisco TMS will only use the Application Integration Package key.

Enabling Option Keys

To enable an option key in Cisco TMS:

1. Go to **Administrative Tools > Configuration > General Settings**.
2. In the **Licenses and Option Keys** pane, click **Add Option Key**.
3. Input the option key string.
4. Click **Save**.

Per System Licensing

Once the per system option key has been activated in Cisco TMS, the **Allow Remote Bookings** setting determines whether each system is using a license.

This setting is void and hidden if the Application Integration Package option is used. If both option keys are added, only the Application Integration Package option will be used by Cisco TMS.

The first time a system is booked through Cisco TelePresence Management Suite Extension Booking API, **Allow Remote Bookings** will be toggled to *Yes* for that system in Cisco TMS, provided a license is available. If no more licenses are available, **Allow Remote Bookings** will be left as *No* for that system, and the requested booking will be denied. A Cisco TMS ticket will be generated to notify the administrator that no more licenses are available.

Note that Cisco TMSXE performs a test bookings as each endpoint is added through the configuration tool, thus also enabling **Allow Remote Bookings**.

To view and/or modify the setting:

1. In Cisco TMS, go to **Systems > Navigator**.
2. Select the system you want.
3. Click the **Settings** tab.
4. In the **TMS Scheduling Settings** pane, you will find *Allow Remote Bookings*.

   If the setting is *Yes*, the system is currently using an Exchange Integration Option license.
5. To disable the setting:
   a. Click **Edit Settings**.
   b. Uncheck *Allow Remote Bookings*.
   c. Click **Save**.

Booking Rights

Importing from Cisco TMS and booking meetings through the API requires authentication with Cisco TMS.

There are two possible models for user authentication.

Service User Books on Behalf of Users

The simplest approach is configuring one service user in Cisco TMS for your client, and granting this user *Book on behalf of* permissions as described below.

This model is only appropriate if all users with access to the client can be granted the same permissions in Cisco TMS, as booking on behalf of someone will create a new Cisco TMS user in the default group if the user does not already exist.

All Users Book for Themselves

If you need to set up different booking rights per user, or block some users from booking altogether, each user must authenticate with Cisco TMS individually through the API.

Setting up Permissions

All users who will book meetings using the API must be members of a group whose permissions include *Booking*. Users who book on behalf of others also require *Book on behalf of* permissions.

Permissions in Cisco TMS are set on a group level. To modify the permissions set for a group:

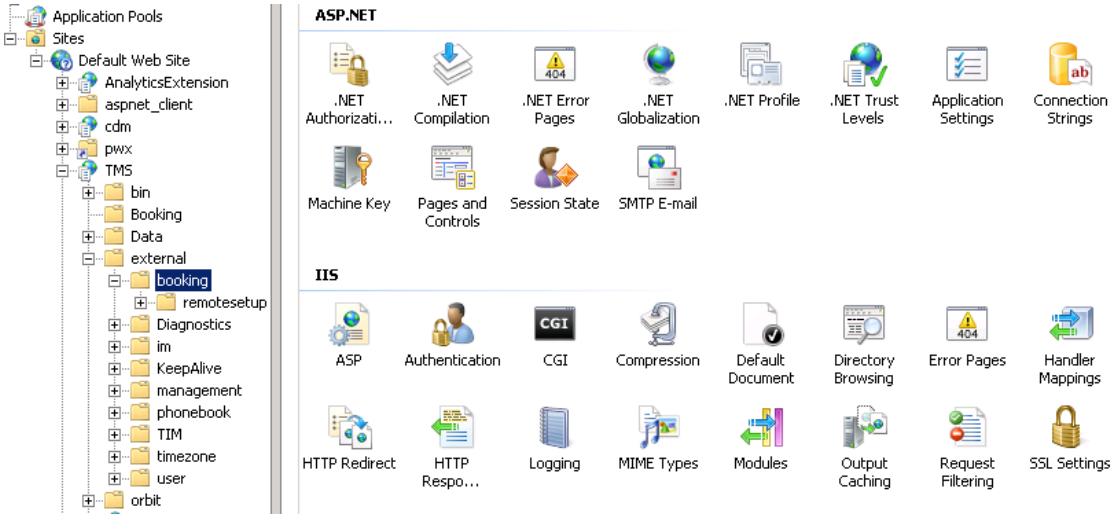
1. Go to **Administrative Tools > User Administration > Groups**.
2. Hover over the desired group, click the drop-down button and select **Set permissions**.
3. In the **Booking** section, under **Misc**, check:
   - *Booking*
   - *Book on behalf of*
   - *Update*
   - *Approve Meeting*, unless bookings are to require manual approval
4. Click **Save**.

NTLM Authentication

On a default Cisco TMS installation, any API request requires the use of Windows Challenge Response or NTLM authentication.

Not all environments support this authentication mechanism (non-Windows based environments), and you therefore may need to allow for Basic Authentication:

1. Open Internet Information Services manager on the Cisco TMS server.
2. Expand **Sites > Default Web Site** and browse to the **/TMS/external/booking** virtual directory.



3. In the **IIS** section, double-click on **Authentication**.
4. Right-click on **Basic Authentication** and select **Enable**.

When using Basic Authentication, we strongly recommend requiring a secure connection using SSL.

Anonymous Authentication is not supported with Cisco TMSBA.

Cisco Collaboration Meeting Rooms Hybrid Requirements

In order to use Cisco TMSBA for meetings that include WebEx, Cisco TMS must be set up with:

- one or more WebEx sites
- WebEx credentials for each user (not service user), either manually added or using WebEx/Cisco TMS single sign-on

For guidance on setting up Cisco Collaboration Meeting Rooms Hybrid with or without single sign-on, see *Cisco Collaboration Meeting Rooms Hybrid Configuration Guide*.

Setting up Your Environment

Cisco TMSBA provides a Web Services API that interfaces with the Cisco TMS booking engine. Web Services allows for simple integration into most common language and programming environments.

See your development tool reference for information on how to build implementation stubs to help speed the development of applications that use Web Services.

API Locations

- The WSDL file for the Cisco TMS remote setup API is located at: **https://[Cisco TMS server] /tms/external/booking/remotesetup/remotesetupservice.asmx**
- The WDSL file for Cisco TMSBA is located at: **https://[Cisco TMS server] /tms/external/booking/bookingservice.asmx**

Microsoft Visual Studio .NET users can reference the APIs by selecting **Project > Add Web Reference**, or entering the URLs above. You will be required to authenticate through web services to create the reference.

In a network load balancing scenario, use the virtual IP address or DNS name of the cluster for this task to allow failover for the API.

Security

We strongly recommend using SOAP for all requests. For security reasons, HTTP GET and POST are only accessible from localhost. To change this, modify the Cisco TMS **web.config** file.

Optional Elements

Optional elements are indicated in different ways in the WSDL:

- `minOccurs = 0` means the element can be omitted from the request XML altogether.
- `nillable="true"` means the element can hold an empty value.

The combination of these two is an element that can be skipped and be set to null (not a value).

Boolean Values

The boolean lexical forms "1" and "true" are interchangeable in SOAP.

The SOAP Header

ExternalAPIVersionSoapHeader

Each call made to the Cisco TMSBA must include the following header specifying the version of the API. The value specified in ClientVersionIn is used by the API to determine the output from the function. The XML below describes the ExternalAPIVersionSoapHeader object that is common for all calls to the API.

Do not set a number greater than the latest version of the API, as this may break compatibility when using later revisions.

```
<ExternalAPIVersionSoapHeader xmlns="http://tandberg.net/2004/02/tms/external/booking/">
       <ClientVersionIn>int</ClientVersionIn>
       <ClientIdentifierIn>string</ClientIdentifierIn>
       <ClientLatestNamespaceIn>string</ClientLatestNamespaceIn>
       <NewServiceURL>string</NewServiceURL>
       <ClientSession>string</ClientSession>
</ExternalAPIVersionSoapHeader>
```

If no version number is set, version *0* is the default.

ClientSession

Calls that require an API integration license must include a ClientSession ID that Cisco TMS will use to recognize the client.

- If the ClientSession ID is null or blank, Cisco TMS will throw an exception containing a suggested ClientSession ID string, and the API call will fail.
- Each ID expires after 47 minutes with a 17 minute added grace period.
- Reusing an expired ClientSession ID will result in Cisco TMS throwing an exception containing a suggested new ID, and the API call will fail.

The client must therefore regularly renew these IDs. We recommend using the suggested ClientSession ID strings from the exceptions thrown by Cisco TMS.

In Cisco TMS the **Active Application Integration Clients** table under **Administrative Tools > Configuration > General Settings > Licenses and Option Keys** lists the network addresses and the ClientSession IDs used by Cisco TMSBA clients that consume Application Integration Licenses. If your Cisco TMSBA client is unable to save conferences

because of ClientSession ID or licensing problems, use this table to identify other clients or Client Session IDs that inadvertently consume licenses.

Refer to your version of Cisco TelePresence Management Suite Administrator Guide for detailed description of the **Active Application Integration Clients** table.

You can enable logging for the ClientSession ID by setting the log level for **log-web-external** to DEBUG. The log will then flag:

- Incoming session IDs
- Number of active session IDs
- Whether the incoming session ID is consuming a license key
- When the number of available licenses is exceeded and the incoming session ID associated with this request

ContextHeader

The context header contains multiple flags related to the conference and the client.

Two flags control whether an email confirmation of the request will be sent out, and whether the confirmation will include conference information such as routing, list of participating systems, and so on. These flags are not mandatory, and all are *false* by default.

The client language flag will request localized API error messages for the specified language. Cisco TMS will return localized messages if available. If no language is specified, English will be returned.

```
<ContextHeader    xmlns="http://tandberg.net/2004/02/tms/external/booking/">
       <SendConfirmationMail>boolean</SendConfirmationMail>
       <ExcludeConferenceInformation>boolean</ExcludeConferenceInformation>
       <ClientLanguage>string</ClientLanguage>
</ContextHeader>
```

GUI Pattern

Cisco TMSBA can supply data to the front-end GUI of an external booking application. There are three information types.

System Information

Information on Cisco TMS resources can be exported to an external application. By using the remote setup API, data on systems in Cisco TMS can be exported to a front-end GUI and used to display system entities available in Cisco TMS.

Get a list of available systems in Cisco TMS by using either:

- `GetSystems`
- `GetSystemsForUser`

These functions return lists of TMSSystem objects, that include information such as the ID of the system, to show in the front-end GUI. `GetSystems` will return all systems in Cisco TMS, while `GetSystemsForUser` will only return the systems the user has booking privileges for. If the external GUI application controls system access, use `GetSystems` and filter the systems in the application.

Free/Busy Information

Free/busy information on systems in Cisco TMS can be exported to an external application using Cisco TMSBA.

- Get all Cisco TMS bookings for a specific user by using `GetConferencesForUser`.
- Get free/busy information for systems by using `GetConferencesForSystems`.
- Get a specific conference, including any exceptions if it is a series, by using `GetRecurrentConferenceById`.

- The Remote Setup API function `GetUsers` returns all users registered in Cisco TMS. The output of this function can be used to display a drop-down list of all users in Cisco TMS, or show conferences booked by a specific person.

Booking Management

The API allows you to forward booking requests from an external booking system to Cisco TMS, and reserve resources in Cisco TMS. (Information exchange: **External Booking System > TMS**).

- Get Conference objects with default values for Conference properties defined in Cisco TMS by using the `GetDefaultConference` function.
- Retrieve existing conferences by using `GetConferenceById`, `GetConferenceIdByExternalId` or `GetRecurrentConferenceById`.
- Save changes to a conference by editing the conference properties and using the function `SaveConferenceWithMode`. This will save the conference to Cisco TMS if the properties validate. If not, an exception will be raised. When saving, you must provide the same version value as you got when retrieving the conference, or the save will be declined. If the save is declined because of your version value is out of date, retrieve the conference again prior to attempting another save operation.
- Delete a conference or series by using `DeleteConferenceById`. Beware that the ongoing conferences will be ended rather than deleted and that conference participants will be disconnected, if the conference is ended while it is active or connected.
- Delete an occurrence of a series by using `DeleteConferenceRecInstanceById`.
- Add recording to a conference by using `GetRecordingAliases` to get information about a user's recording aliases and use this information to add recording participant(s) to the conference.

Replication Pattern

The APIs can be used in conjunction with external booking applications that have their own reservation database. There are three main components.

Import

The API can automate importing systems from Cisco TMS into a third-party application, or this can be initiated by the user via a GUI.

Get a list of available systems in Cisco TMS by using either of the following from the remote setup API:

- `GetSystems`
- `GetSystemsForUser`

These functions return lists of TMSSystem objects, and information such as the ID of the system, for use by a third party application. `GetSystems` will return all systems in Cisco TMS, while `GetSystemsForUser` will return only the systems the user has booking privileges for. If the external application controls system access, use `GetSystems` and filter the systems in the application.

Replication

External booking systems can keep track of booking transactions on the Cisco TMS server, and replicate bookings made using Cisco TMS. This part does not apply to external GUI front ends that do not have their own reservation database.

Get a list of transactions listed by transaction ID by using the GetTransactionsSince function. All conferences have a transaction ID property

The list of transactions contains:

- The transaction type (*New*, *Update*, and *Delete*)
- An associated conference ID.

Use `GetConferenceById` to get an updated Conference object and update the conference with the external source. The current transaction ID should then be updated to the last conference's TransactionId.

Booking

The API allows you to forward booking requests from an external booking system to Cisco TMS, and reserve the resources there.

- Get Conference objects with default values for Conference properties defined in Cisco TMS by using `GetDefaultConference`.
- Retrieve saved conferences by using one of the following functions:
  - `GetConferenceById`
  - `GetConferenceIdByExternalId`
  - `GetRecurrentConferenceById`
  - `GetConferencesForUser`
  - `GetConferencesForSystems`
- Save changes to a conference by editing the conference properties and using the function `SaveConferenceWithMode`. This will save the conference to Cisco TMS if the properties validate. If not, an exception will be raised. When saving, you must provide the same Version value as you got when retrieving the conference, or the save will be declined. If the save is declined because your Version value is out of date, retrieve the conference again prior to attempting another save operation

Availability

The API allows you to display reservations from the Cisco TMS internal reservation database.

Limitations

WebEx Booking Limitations

WebEx booking does not support all recurrence patterns and options supported by Cisco TMS:

- Exceptions from a recurrence pattern (moving or updating one or more meetings) are not supported.
- Exceptions from a recurrence pattern (moving or updating one or more meetings) are not supported, unless WebEx is booked as **OwnedExternally**. For more details see OwnedExternally, page 32
- Some advanced recurrence patterns are not supported.

When WebEx is not supported for a particular booking, the booking will fall back to telepresence only.

WebEx-only meetings not supported

WebEx-only meetings should not be booked using Cisco TMSBA. If using Cisco TMS to create a conference that includes WebEx without including any telepresence participants, telepresence resources will still be taken up, as an MCU will connect to the conference. The API will book the conference, but return a warning when such conferences are created.

Also note that WebEx bookings do not support time zone rules.

Other Booking Limitations

The following limitations apply when booking through Cisco TelePresence Management Suite Extension Booking API:

The following limitations apply when booking through Cisco TMSBA or any other extension using Cisco TelePresence Management Suite Extension Booking API:

- Cascading to additional MCUs when the number of participants exceeds the capacity of the first MCU is not supported.
  To support such scenarios, set up Cisco TelePresence Conductor as the preferred MCU in Cisco TMS.
- When a service user is performing all bookings, the booking permissions are the same for all users. Individual permissions and restrictions in Cisco TMS are ignored.
- Meetings in the past cannot be changed or deleted, and you cannot move a meeting from the past to the future.
- If sufficient system licenses are not available at the time of editing an existing booking, the booking will be deleted.
- Yearly recurrence is not supported.

Booking Horizon and Recurrence

Cisco TMS will decline any meeting request that is not within its booking horizon or that has an unsupported recurrence pattern:

- Series with more than 100 occurrences or with no end date.
- Meetings including occurrences outside of the Cisco TMS booking window. We strongly recommend configuring identical booking windows for Cisco TMS and all integrated resource mailboxes in Exchange.
- Meetings in the past.

Ongoing Meetings

Updating a single meeting that is currently ongoing is possible, but will not always be successful.

- Modifying any meeting, extending the meeting will fail if it creates a booking conflict for any of the participants.
- Modifying single meetings, including meetings that are part of a series:
  - Editing the start time will not work and Cisco TMS will throw an exception.
  - Editing the meeting so that it would be required to be disconnected and re-routed will not be successful. For example, if the meeting is using a bridge that does not support WebEx, you cannot add WebEx during the meeting.
  - Any other aspects of the meeting can be modified, but if the number of participants exceeds the available capacity of the MCU or TelePresence Server, Cisco TMS will throw an exception and the participants will not be added.
- *Deleting* a recurrent series while a meeting in the series is ongoing will cause the ongoing meeting to end.
- *Modifying* a recurrent series while a meeting in the series is ongoing will turn the ongoing occurrence into a single meeting, separate from the series:
  - Any occurrences of the modified series that are in conflict with the ongoing meeting, will not be created.
  - Any past occurrences in the series will not be modified.
  - Pending occurrences are assigned new conference IDs.
- *Modifying* a conference when Extend mode is set to Endpoint Prompt or Automatic Best Effort in Cisco TMS:
  - If the conference is modified during the first extension (changes except End time modification), the endpoints will receive the meeting extension message, but conference extension pop up/message will be displayed again at the endpoints
  - Modifying the conference again after extension, may end the conference.

Unused Values

The following values are present in the API, but not in use:

Function	Attribute	Value
RecurrencePattern	PatternEndType	*Never*
RecurrencePattern	FrequencyType	▪ *Secondly* ▪ *Minutely* ▪ *Hourly* ▪ *Yearly*
Participant	ParticipantCallType	*User*
ISDNBandwith	Bandwidth	*Max*
IPBandwith	Bandwidth	*Max*

Remote Setup API Reference

The remote setup API accommodates the setup of users and systems prior to using the Booking API.

TMSSystem Object 18
TMS User Object 20
Remote Setup API Functions 21

TMSSystem Object

The TMSSystem object contains information about a system in Cisco TMS. This object is used to read information, as the remote setup API does not support updating system information in Cisco TMS.

Use this object to import the required information into the third party application. The **SystemId** is required to connect the application entity with the system in Cisco TMS. In addition other information can be imported and shown for informative purposes, for example the name of the system.

TMSSystem

SystemId	The ID of the system in Cisco TMS. Use this to refer to the associated system in Cisco TMS from your application. For example, when booking a conference, insert the IDs of the chosen systems into the Conference object.
SystemName	The name of the system in Cisco TMS. Use this to display the name of the system in your application.
Contact	The system contact associated with the system in Cisco TMS.
Manufacturer	The manufacturer of the system. For example, Cisco.
Description	A textual description stored in Cisco TMS. This field can contain information such as the number of chairs in the meeting room where the system is located.
SystemType	The type of system. For example, Cisco TelePresence System EX90.
ISDNNumber	The ISDN number of the system.
Location	The ISDN location where the system is located.
NetworkAddress	The fully qualified ISDN number of the system. A fully qualified ISDN number always includes the country code and area code. This is not implemented.
WebInterfaceURL	The http address of the web server of the system.
SIPUri	The SIP URI of the system.
H323Id	The H.323 ID of the system.

E164Alias	The E.164 alias of the system.
TimeZone	The time zone where the system is located.
SystemCategory	The system category.
SystemStatus	The status of the system.

TimeZone

TimezoneName	The name of the time zone.
StartTimeDTS	The start date of daylight saving time.
EndTimeDTS	The end date of daylight saving time.
GMTOffset	The GMT (UTC) offset.

SystemCategory

SystemCategory	An enumeration value of what category of system this is. The available options are: ■ *Endpoint* ■ *Equipment* ■ *Room* ■ *Recording*

SystemStatus

SystemStatus	An enumeration with the status of the system when this function is call. Note that the status of the system can change frequently. The possible values are: ■ *Alive* ■ *Idle* ■ *InCall* ■ *NoResponse* ■ *Unknown*

TMSSystem Object XML

The XML below describes the TMSSystem object. Following the XML is a description of the elements and what information each element it contains.

Note that as all fields are not required, the output may contain less system information than the object can hold.

```
<TMSSystem>
	<SystemId>long</SystemId>
	<SystemName>string</SystemName>
	<Contact>string</Contact>
	<Manufacturer>string</Manufacturer>
	<Description>string</Description>
	<SystemType>string</SystemType>
	<NetworkAddress>string</NetworkAddress>
```

```
        <Location>string</Location>
        <ISDNNumber>string</ISDNNumber>
        <QNumber>string</QNumber>
        <WebInterfaceURL>string</WebInterfaceURL>
        <SIPUri>string</SIPUri>
        <H323Id>string</H323Id>
        <E164Alias>string</E164Alias>
        <TimeZone>
                <TimezoneName>string</TimezoneName>
                <StartTimeDTS>string</StartTimeDTS>
                <EndTimeDTS>string</EndTimeDTS>
                <GMTOffset>string</GMTOffset>
        </TimeZone>
        <SystemCategory>
                <systemCategory>Endpoint or Equipment or Room or Recording</systemCategory>
        </SystemCategory>
        <SystemStatus>
                <SystemStatus>Alive or Idle or InCall or NoResponse or Unknown</SystemStatus>
        </SystemStatus>
</TMSSystem>
```

TMS User Object

The Cisco TMS user object contains information about Cisco TMS users. Use this object to access information about users in Cisco TMS. The XML document below describes the User object. Following the XML is a description of the elements and what information each element it contains.

User

Attribute	Description
DisplayName	The display name of the user.
EmailAddress	The e-mail address of the user.
FirstName	The first name of the user.
LastName	The last name of the user.
UserName	The Windows login name of the user.
IsHiddenUser	Boolean value used to represent whether this is a normal user (True), or a service account (False) that normally should not be displayed in a list of users.
TimeZone	The time zone where the user is located. Uses the same TimeZone object as TMSSystem.

TMS User Object XML

```
<User>
    <DisplayName>string</DisplayName>
    <EmailAddress>string</EmailAddress>
    <FirstName>string</FirstName>
    <LastName>string</LastName>
    <UserName>string</UserName>
    <IsHiddenUser>boolean</IsHiddenUser>
    <TimeZone>
        <TimezoneName>string</TimezoneName>
        <StartTimeDTS>string</StartTimeDTS>
        <EndTimeDTS>string</EndTimeDTS>
        <GMTOffset>string</GMTOffset>
    </TimeZone>
</User>
```

Remote Setup API Functions

This reference section describes all the available functions of the remote setup API.

DisableConferenceAPIUser

This function is used to disable a ConferenceAPI user. E-mail notifications for the user are disabled, and the user is removed from all groups in Cisco TMS except the Users group. This is done to keep references valid. Executing this function requires Cisco TMS Site Administrator privileges.

This function is typically used during uninstall procedures.

Supported parameters:

`userName`	The full username of the user to delete in NT4 style (domain\username).

Returned data: None.

GenerateConferenceAPIUser

This function generates a Cisco TMS Booking API account in the default user container on the Cisco TMS server, including registering the user in Cisco TMS. It is typically used during installation/setup procedures.

The user will be:

- Hidden from normal user lists.
- Added to the Site Administrator Group.

In order for the function to complete, the current user must be:

- A Cisco TMS Site Administrator
- A local computer administrator

Supported parameters:

`userNameBase`	The base portion of the user name. If a user with the name already exists a numeric postfix is added (for example tms-admin ==> tms-admin1).
`encPassword`	A base64 encoded password that is to be used for the newly created user.
`emailAddress`	The email address of the user.
`sendNotifications`	Whether the user should receive scheduling notifications.

Returned data: The username of the created user (NT4 domain/username style).

GetConferenceLanguages

Returns an array of Language objects.

Supported parameters: None

Returned data: An array of supported conference languages in TMS. The CultureInfo field specifies the exact variety of a language and can be used to set the ConferenceLanguage on the Conference object when scheduling conferences.

GetSystemById

This function returns information about a specific system. If the system is not found, this causes an error.

Supported parameters:

`TMSSystemId`	System ID as given in Cisco TMS.

Returned data: A TMSSystem object.

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetSystems

This function returns all endpoints and rooms registered in Cisco TMS. Note that network systems, such as a Cisco TelePresence MCU, are not returned since they are normally not booked by the users, but are added to the conference by Cisco TMS if required.

Typically used during setup of resources in the external booking system to connect resources in Cisco TMS with resources in the external booking system.

Supported parameters: None

Returned data: An array of TMSSystem objects.

GetSystemsForUser

This function returns all endpoints and rooms that can be booked by the current user, the account credentials are used to communicate with Cisco TMSBA.

Note that network systems, such as a Cisco TelePresence MCU, are not returned since they are normally not booked by the users, but are added to the conference by Cisco TMS if required.

Typically used in the external booking system to list Cisco TMS resources in external booking system.

Supported parameters: None.

Returned data: An array of TMSSystem objects.

GetUsers

This function returns all users registered in Cisco TMS.

This function is typically used in the front-end GUI to provide a list of Cisco TMS users, and can filter output from the Cisco TMSBA based on users from this output.

Supported parameters: None.

Returned data: An array of User objects.

IsAlive

This function checks the connection to the web services of Cisco TMS.

It is typically used during installation to check the URL to this web service.

Supported parameters: None

Returned data: A boolean value true/false, which is *true* if the connection works.

IsBookOnBehalfOfUser

Checks whether the specified user (not the current user) is a member of a Cisco TMS group that has permissions to book on behalf of other users.

Supported parameters:

`user`	The Cisco TMS user ID of the user for whom to check permissions.

Returned data: A boolean value true/false, which is *true* if the user if the user has permissions to book on behalf of other users in Cisco TMS.

IsLocalAdmin

This function checks whether the current user can create local or Active Directory accounts in the default user container on the Cisco TMS server.

This is typically used during installation to check whether the user installing the integration has sufficient access to Active Directory and the Windows server hosting Cisco TMS.

This function must return *True* in order for the GenerateConferenceAPIUser function to succeed.

Supported parameters: None.

Returned data: A boolean value true/false, which is *true* if the user is a local administrator.

IsTMSBookOnBehalfUser

Checks whether the current user is a member of a Cisco TMS group that has permissions to book on behalf of other users.

Supported parameters: None.

Returned data: A boolean value true/false, which is *true* if the user has permissions to book on behalf of other users in Cisco TMS.

IsTMSServiceUser

This function is used to check whether the current user is flagged as an Exchange Integration user and has access to book on behalf of other users. The service user setting has been deprecated, and this function will be removed in a future version of Cisco TMS.

This is typically used during installation to check whether the user installing the integration has sufficient access permissions for the Cisco TMS server.

Supported parameters: None.

Returned data: A boolean value true/false, which is *true* if the user is a Cisco TMS service user.

IsTMSSiteAdmin

This function checks whether the current user is a member of the Cisco TMS Site Administrators group.

This is typically used during installation to check whether the user installing the integration has sufficient permissions towards the Cisco TMS server. This function should return *True* for the `GenerateConferenceAPIUser` function to succeed.

Supported parameters: None.

Returned data: A boolean value *true*/*false*, which is *true* if the user is a Cisco TMS Site Administrator.

SetPrimarySystem

Used to set a specific endpoint as a primary system for the logged-in user.

Supported parameters:

`primSys`	A Cisco TMS system ID.

Returned data: A boolean value true/false, which is *true* if the primary system was successfully set for the current user.

Booking API Reference

The booking API lets you schedule conferences in Cisco TMS using a third-party client, and replicate existing bookings between the two. This chapter is a reference to the conference object and to all available functions and parameters.

Conference Object 24
Booking API functions 43

Conference Object

This object can be used to read and write:

- Conference properties such as **Start Time**, **End Time**, **Conference Title**, and **Conference Password**.
- Conference call-related values such as **Bandwidth**, **Picture mode**, and **Encryption mode**.

All conference resources, including video participants, audio participants, phone book participants, external participants and so on, are held in this object, together with routing information for connecting the resources. You also use the Conference object to define the conference type, that is, how the conference should be connected.

Conference data can be saved/updated, and handled by Cisco TMS using the `SaveConferenceWithMode` function described below.

Conference

Attribute	Read/Write	Description
ConferenceId	r/w, optional	When using **SaveConferenceWithMode**: ▪ Set to `-1` to create a new conference using **GetDefaultConference**. ▪ Set to a value greater than 0 to update the existing conference that has the given ID in the Cisco TMS database.
Title	r/w, optional	If no title is specified, the Cisco TMS default will be used. Title values can not exceed 255 characters. Note: Single and double quotes in the **Title** field will be discarded.
StartTimeUTC	r/w, required	The start and end times of the conference in UTC format. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.
EndTimeUTC	r/w, required	

Attribute	Read/Write	Description
RecurrenceInstanceIdUTC	r, only used when getting conference from Cisco TMS	Gives the start time of the instance of the meeting according to the recurrence pattern. If this is different from **StartTimeUTC**, the meeting is an exception to the recurrence pattern. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.
RecurrenceInstanceType	r, only used when getting conference from Cisco TMS	If this string contains the value 'modify' it means that the particular meeting is an exception to a recurrence pattern. It the string contains " deleted" , it is a meeting that has been deleted from a series of recurring meetings.
FirstOccurrenceRecInstanceIdUTC	r, only used when getting conference from Cisco TMS	Gives the start time of first instance of the meeting according to the recurrence pattern. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.
RecurrencePattern	r/w, optional	Sets the recurrence patterns for recurrent meetings. This is not valid if you call the `SaveConferenceRecInstance` function. See RecurrencePattern, page 39.
OwnerId	r/w, optional	Elements used to look up the owner of the conference: ■ If **OwnerId** is a valid Cisco TMS user id number, it will be looked up in the Cisco TMS database and that value is used. **OwnerUserName** and **OwnerEmailAddress** are not considered. ■ If **OwnerId** is set to `-1`, then **OwnerUserName** is considered and **OwnerEmailAddress** is not considered. ■ If **OwnerId** is not specified, then **OwnerEmailAddress** will be looked up in Cisco TMS. If not found and Active Directory Lookup is enabled, a lookup will be performed. If no username is found, Cisco TMS will create a username based on the email address used.
OwnerUserName	w, optional	
OwnerEmailAddress	w, optional	
OwnerFirstName	w, optional	These elements are not used for lookup, but can be stored for the new user if it does not exist in Cisco TMS and Active Directory Lookup is not enabled.
OwnerLastName	w, optional	

Attribute	Read/Write	Description
ConferenceType	r/w, optional	Setting determining how the conference will be launched. The valid values are: ■ *Automatic Call Launch*, connect all participants at conference start time and disconnect them again at conference end time. ■ *One Button to Push*, allows for OBTP call setup on supported systems. ■ *Manual Call Launch*, the conference master participant will be asked to connect the call at conference start time. ■ *No Connect*, reserve the participants and generate the call route, but do not automatically connect the conference. ■ *Reservation Only*, reserve the participants for the conference duration, but do not create a route. ■ *Default*, do not change the conference type currently specified for the conference in Cisco TMS. If unspecified, the default conference type configured in Cisco TMS will be used.
Bandwidth (Discontinued)	–	This item has been deprecated and is included for backwards compatibility only. Use **ISDNBandwidth** and **IPBandwidth** to control the conference bandwidth. See ISDNBandwidth, page 41 and IPBandwidth, page 41.
PictureMode	r/w, optional If not specified, Default is assumed	The valid values are: ■ *Continuous Presence* ■ *Enhanced CP* ■ *Voice Switched* ■ *Default*, do not change the picture mode currently specified for the conference in Cisco TMS. If unspecified, the default conference type configured in Cisco TMS will be used. This setting does not apply to Cisco TelePresence MCU Series, Cisco TelePresence Server and TelePresence Conductor

Attribute	Read/Write	Description
Encrypted	r/w, optional If not specified, Default is assumed	The encryption mode for the conference. The valid values are: ■ *Yes* ■ *No* ■ *If Possible* ■ *Default*, do not change the picture mode currently specified for the conference in Cisco TMS. If the meeting is created using Cisco TelePresence Management Suite Extension Booking API that has external participants, then set the encryption mode to "*If Possible*" to create a successful booking. If unspecified, the default encryption setting configured in Cisco TMS will be used.
DataConference	r/w, optional	Legacy method of adding WebEx to a conference. See **ExternalConference** below for the preferred method. The valid values are: ■ *Yes* ■ *No* (default) ■ *If Possible* ■ *Default*, do not change the data conference setting currently specified for the conference in Cisco TMS.
ExternalConference	r/w, optional	Used to include a WebEx conference with the telepresence meeting, or a dial-in address for an Externally Hosted Conference. See ExternalConference, page 31.
EmailTo	r/w, optional	Corresponds to the "Send Email To" field in Cisco TMS conference information. May be used for one or more email addresses that will receive conference information from Cisco TMS. Email addresses can be separated by semicolons, commas, or spaces.
ShowExtendOption	r/w, optional If not specified, Default is assumed	Set this value to specify extend option behavior when the scheduled conference is close to ending. ■ *Yes*–prompt the VC Master (the participant with a ParticipantCallType of TMS Master Participant) to extend the conference ■ *No*–do not offer to extend conference ■ *Automatic Best Effort*–automatically extend conference ■ *Default*–do not change the **Show Extend Option** setting currently specified for the conference in Cisco TMS If unspecified, the default Show Extend Option defined in the **Administrator Tools** Page in Cisco TMS is used.

Attribute	Read/Write	Description
Password	r/w, optional	The numeric PIN that conference participants must enter to join the call. If **Auto Generate PIN on New Conferences** is enabled in Cisco TMS, a random PIN will be added to any conference that does not have a PIN specified.
BillingCode	r/w, optional	The billing code to use for the conference. If Cisco TMS requires billing codes, this field must be specified and will be validated against the list of billing codes in Cisco TMS. If no match is found, the conference will not be created and the API will throw an "Invalid billing code" exception. This setting is blank by default.
ISDNRestrict	r/w, optional	Whether the ISDN channels should be restricted to use 54k and not 64k. The default is *No*.
ConferenceInfoText	r, only used when getting conference from Cisco TMS.	Information on how the conference will connect, including routing information. Based on Cisco TMS templates.
ConferenceInfoHTML	r, only used when getting conference from Cisco TMS.	Information in HTML markup on how the conference will connect, including routing information. Based on Cisco TMS templates.
UserMessageText	r/w, optional	A user definable text/description of the conference. The default is blank.
ExternalSourceId	r/w, optional	An external source and ID definable by the client, this is used to synchronize the Cisco TMS database with the external sources database. If Cisco TMS is given a value for these fields, Cisco TMS will return the value for all instances of the same conference. Using "TMS" and "TMS-ADHOC" as external source IDs is reserved for Cisco TMS internal use. Cisco TMSBA will throw an exception if either is used by another client. The default for these two is blank.
ExternalPrimaryKey		
DetachedFromExternalSourceId	r/w,optional	When an instance of a series has been detached, these fields will contain the **ExternalSourceId** and **ExternalPrimaryKey** of the original series.
DetachedFromExternalPrimaryKey		

Attribute	Read/Write	Description
Participants	r/w, required	A list of conference participants. When calling `GetDefaultConference`, the participant list will be empty. Note that when updating an occurrence of a series: ▪ Setting this to *null* will leave the participants unchanged. ▪ Setting **Participants** to an empty list will clear all participants.
RecordedConferenceUri	r, only used when getting conference from Cisco TMS	If the conference is recorded, this is the URI of the conference recording.
WebConferencePresenterUri	r, only used when getting conference from Cisco TMS	The URIs that the presenter and attendees respectively will use to join WebEx when the legacy DataConference attribute has been used. See ExternalConference for the recommended way to add WebEx.
WebConferenceAttendeeUri	r, only used when getting conference from Cisco TMS	

Attribute	Read/Write	Description
ISDNBandwidth	r/w, optional	The ISDN and IP bandwidths of the conference. If unspecified, the default ISDN bandwidth configured in Cisco TMS will be used. The valid values are: ■ *1b/64kbps* ■ *2b/128kbps* ■ *3b/192kbps* ■ *4b/256kbps* ■ *5b/320kbps* ■ *6b/384kbps* ■ *8b/512kbps* ■ *12b/768kbps* ■ *18b/1152kbps* ■ *23b/1472kbps* ■ *30b/1920kbps* ■ *32b/2048kbps* ■ *48b/3072kbps* ■ *64b/4096kbps* ■ *7b/448kbps* ■ *40b/2560kbps* ■ *96b/6144kbps* ■ *Max*, use as much bandwidth as available and necessary. ■ *6000kbps* ■ *Default*: Use Cisco TMS default for a new conference. If updating an existing conference, *Default* will not change the bandwidth currently specified for the conference in Cisco TMS.
IPBandwidth	r/w, optional	
ConferenceLanguage	r/w, optional If not specified, default is assumed	The language used for email invitations and other notifications regarding the meeting. A list of supported languages is available from the Remote Setup API, see GetConferenceLanguages, page 21.
ConferenceTimeZoneRules	r/w, optional	An array of time zone rules for the conference. See ConferenceTimeZoneRules, page 35.

Attribute	Read/Write	Description
ConferenceState	r/w, optional	Contains the **Status** element, for which the valid values are: ■ *All* ■ *AllExceptDeleted* ■ *Pending* ■ *Ongoing* ■ *Finished* ■ *PendingAndOngoing* ■ *MeetingRequest* (Conference has been requested by a user without booking rights, needs approval.) ■ *Rejected* ■ *NotSaved* ■ *Defective* ■ *Deleted*
Version	r/w, optional	Conference revision number set, incremented, and returned by Cisco TMS with each save. Defaults to 0 if not included by the client. If the client tries to update a version that is not the latest in the database, Cisco TMS will throw an exception.
Location	r/w, optional	A textual description of the physical location(s) of the conference.
Invitees	r/w, optional	A semi-colon separated list of invitees' email addresses.
MeetingSource	r/w, optional	Source of the external meeting can be identified. Using "EX:PROXY" as MeetingSource is reserved for Cisco TMS internal use.

ExternalConference

The ExternalConference element is used to include WebEx with telepresence conferences in CMR Hybrid deployments, and for externally hosted conferences, that is, conferences hosted on a system outside of Cisco TMS.

ExternallyHosted

An Externally Hosted Conference is a conference that has been created outside of your Cisco TMS. For example, if another company is hosting a conference and has provided a dial-in video address, you can schedule endpoints in your organization by booking an Externally Hosted Conference in Cisco TMS, making the endpoints dial in to the conference as *One Button To Push* or *Automatic Connect* if desired.

Making a conference externally hosted limits the set of available booking and monitoring features, as Cisco TMS does not control the conference host in any way.

The following Cisco TMS booking features and participants cannot be used with an Externally Hosted Conference:

- WebEx
- MCUs
- Guaranteed encryption (**Secure** cannot be set to *Yes*)
- Recording participants

- Non-SIP dial-in participants
- Dial-out participants

Setting **Picture Mode** or **Extend Mode** will also not affect an Externally Hosted Conference.

When monitoring an Externally Hosted Conference in Conference Control Center, features are similarly limited. From CCC, you can add participants and change the end time for the Cisco TMS participants. At end time, these participants will be disconnected and the Externally Hosted Conference will be listed as finished, but Cisco TMS cannot disconnect or extend the Externally Hosted Conference itself.

DialString

The ExternalHost element contains the DialString element, which must be populated with a SIP URI. Cisco TMS will perform no validation of this string, as it is assumed to be a valid video address for a conference set up outside of Cisco TMS.

Attribute	Read/Write	Description
DialString	r/w, optional	The SIP URI used for the externally hosted conference

WebEx

There are two ways to add WebEx to a telepresence booking:

- The recommended method:
  - Set **ExternalConference** with an empty **WebEx** element inside of it to initiate the addition of WebEx to a booking, or an update to existing WebEx information for a conference.
  - The WebEx element must be either empty or contain only the optional **MeetingPassword** attribute. All other attributes are intended for WebEx output only and will be overwritten if set in the booking request.
- The legacy way:
  - Set **DataConference** to *Yes* or *If possible*.
  - Do not set **ExternalConference**.

To remove WebEx from a booking:

- Set an empty **ExternalConference** (no **WebEx** element).
- The legacy way: Set **DataConferenceMode** to *No*, do not include **ExternalConference**.

WebEx exceptions

As of WebEx Meeting Center version WBS29, WebEx does not support exceptions. Exceptions are meetings in a series that diverge from the series pattern. If attempting to use Cisco TMSBA to create a series with exceptions that includes WebEx, the WebEx calendar may become out of sync with Cisco TMS.

WebEx also cannot be added to a single instance of a recurrent series, or removed from a single instance of a series.

WebEx exceptions are supported in Cisco TMSBA if booked with the **OwnedExternally** flag, if the Cisco TMSBA client announces API version 16 or above. This Cisco TMSBA feature is in preparation for exception support in a future WebEx release. See Cisco Collaboration Meeting Rooms (CMR) Hybrid Release Notes for your version of Cisco TMS and WebEx Meeting Center for updates on WebEx support for exceptions to recurring meeting series.

OwnedExternally

The **OwnedExternally** attribute of the WebEx object controls whether the WebEx meeting was originally booked by an external client. This attribute is primarily intended for use by Cisco TMSXE and other Cisco products, but could also be used by other Cisco TMSBA clients when scheduling Cisco CMR Hybrid meetings.

Cisco Cisco TMSBA clients that book with **OwnedExternally** set to *True* are responsible for integrating with the WebEx cloud on their own. The client must first schedule a meeting in the WebEx cloud using the WebEx APIs, and then provide the WebEx details (such as the *SiteUrl*, *HostKey*, and other attributes) returned by the WebEx API to Cisco TMSBA when scheduling the telepresence part of the Cisco CMR Hybrid meeting. When **OwnedExternally** is set *True*, Cisco TMS will attempt no validation of the provided WebEx data, as this is expected to be the responsibility of the client.Cisco TMS will only reserve a conference bridge for the meeting, and instruct the bridge to dial out to WebEx (use the dial string specified by the client in the *SipUrl* element) at the scheduled start time.

When booking with **OwnedExternally** set *True*, Cisco TMSBA allows WebEx to be added to or removed from single instances of recurrent series, as well as moving instances of Cisco CMR Hybrid –enabled recurrent series in time. For conference series where WebEx is **OwnedExternally**, Cisco TMSBA also supports providing different WebEx data for individual instances of the series, such as changing some instances to use a different WebEx site.

See WebExState, page 34 for details on how to enable or disable WebEx for individual instances of a recurrent series.

Attribute	Read/Write	Description
MeetingKey	r/w, optional	The WebEx meeting number.
SipUrl	r/w, optional	The conference SIP URL.
ElementsToExclude	r/w, optional	Elements to exclude from conference invitation email. The following elements may be excluded: ▪ None ▪ MeetingPassword ▪ HostKey ▪ LocalCallInTollFreeNumber ▪ GlobalCallInNumberUrl
MeetingPassword	r/w, optional	The password required to join the WebEx conference. Must comply with WebEx password rules. If the booking does not include a password, WebEx will generate one.
JoinMeetingUrl	r/w, optional	The URL that attendees will use to join the conference.
HostMeetingUrl	r/w, optional	The URL that the host will use to join the conference.
HostKey	r/w, optional	Specifies the host key, which may be used to pass and reclaim the host role during a WebEx conference.
JoinBeforeHostTime	r/w, optional	Whether participants are allowed to join the conference up to 15 minutes before the scheduled start time. The required format is HH:MM:SS.
Telephony	r/w, optional	Phone numbers for joining the conference. The type is **WebExTelephony**, see below.
TmsShouldUpdateMeeting	r/w, optional	Whether a WebEx meeting originally booked by an external client may be updated by Cisco TMS. If updating the meeting is not possible, the request will return a warning.
SiteUrl	r/w, optional	URL of the WebEx site booked for the conference. This site must already be added to Cisco TMS, or the request will return an error.

Attribute	Read/Write	Description
UsePstn	r/w, optional	Whether the WebEx site for the conference is set up to use PSTN. When the conference is externally owned (see below), this setting will always be read/overwritten from the WebEx site configured in Cisco TMS.
OwnedExternally	r/w, optional	Whether the WebEx meeting was originally booked by an external client. See OwnedExternally, page 32 for details.
Errors	r, optional	Errors from WebEx.
Warnings	r, optional	Warnings from WebEx.

WebexTelephony

Attribute	Read/Write	Description
LocalCallInTollNumber	r/w, optional	Specifies the backup toll call number.
LocalCallInTollFreeNumber	r/w, optional	Specifies the toll-free call in number.
GlobalCallInNumberUrl	r/w, optional	Specifies a call-in number for out-of-country participants.
PstnDialInNumber	r/w, optional	Number to dial in if using PSTN. This element is required if **UsePstn** is *true*.
DtmfSequence	r/w, optional	PSTN only. This is a WebEx format, do not convert. This element is required if **UsePstn** is *true*.
ParticipantAccessCode	r/w, optional	The access code used by attendees when dialling in for an audio-only conference.

For further detail on WebEx configuration and options, see your WebEx Site Administrator's User Guide or online help, linked from your WebEx site.

WebExState

The **WebExState** element describes exceptions to a conference series WebEx data. Its purpose it to allow a Cisco TMSBA client not to receive redundant WebEx data for every instance to a series when getting conferences from Cisco TMS, and to add, update or delete WebEx on an instance basis in a non-verbose way. Note that the **WebExState** element is versioned, so that Cisco TMSBA clients announcing version 15 or below cannot use it.

For Cisco Collaboration Meeting Rooms Hybrid conferences, a **WebExState** element exist inside the *ExternalConference* element, both on a series level and on an instance level. Instance level **WebExState** elements reside within the `RecurrencePattern.Exceptions` array used in the *GetRecurrentConferenceById* and *SaveConferenceWithMode* functions, while the series level **WebExState** element resides in the container conference object.

When getting conferences from Cisco TMS using **GetRecurrentConferenceById**, instances that have the same WebEx data as the series itself will have their *WebExInstanceType* attribute set to *Normal*.

To delete WebEx from an instance of a recurrent Cisco CMR Hybrid series, update the *WebExInstanceType* attribute to *Delete* for that specific instance before saving the series. To either add WebEx to an instance of a recurrent series or to update the existing WebEx data for it, set the *WebExInstanceType* attribute to *Modify* for the instance, and provide a full WebEx element for it.

Attribute	Read/Write	Description
WebExState	r/w, optional	Contains the *WebExInstanceType* element, for which the valid values are: ■ *Normal* : the instance WebEx data is the same as the series WebEx data ■ *Modify*: the instance WebEx data is different from the series WebEx data ■ *Delete*: there is no WebEx data for the instance.

ConferenceTimeZoneRules

See Time Zone Versioning, page 7 for information on how time zones are handled with versions of the API that do not support the model described below.

Booking

When booking a new conference using API version 11 or later, Cisco TMS will:

- Use the time zone rules supplied with the booking, if available and valid.
- Use the time zone of the conference owner in Cisco TMS if no time zone rules are supplied with the booking.

Note that each booking, be it for a single meeting or a series, may only have one set of time zone rules. If different rules are specified for start time and end time, the time zone rules for end time will be ignored.

Updating

When updating a conference:

- Setting the time zone to *null*, once set, is not a valid option. Cisco TMS will keep the original time zone of the conference, even if the conference is being modified by a client or API version that does not support time zone rules.
- Supplying the new time zone rule as part of the Conference object will change the actual time zone of this conference.

Note that modifying the time zone of an existing meeting is only supported through the API, not in the Cisco TMS user interface.

TimeZoneRule

ConferenceTimeZoneRules contains an array of **TimeZoneRule** elements. If a change to DST rules is scheduled for a time zone, the new rule set should be included as early as possible to ensure that conferences scheduled to happen after the rule change will happen at the intended time.

Note that no rules can overlap in time, and no start day can be the same day as the end day of the previous rule.

Each TimeZoneRule element may contain the attributes below.

Attribute	Read/Write	Description
ValidFrom		Use SOAP dateTime to specify the exact time from which the rule is valid. Here, you must use the given time zone's standard time and provide it as a datetime of the unspecified kind, or an exception will be thrown. Do *not* use UTC or UTC formatting.

Attribute	Read/Write	Description
Id	optional	Time zone rule set name. The **Id** must be the same for each TimeZoneRule in the ConferenceTimeZoneRules array. If they are not the same, the last **Id** will be used.
DisplayName		Time zone rule set description that will be displayed in Cisco TMS for any booking using this rule. Note that there is no mechanism for localizing this, the textual description will be used as-is.
BaseOffsetInMinutes		UTC offset outside of daylight savings time (DST), in minutes. The offset must not exceed 14 hours/-14 hours.
DaylightOffsetInMinutes		Offset from the base offset during DST, in minutes. The sum of this offset and the base offset must not exceed 14 hours/-14 hours. . If set to *0*, Cisco TMS will not parse the defined Daylight or Standard rules, but create a rule without DST changes
Daylight	optional	Set for any time zone that has rules for entering (Daylight) and leaving (Standard) DST. The type is **TimeChange**, see below. If either of these fields is undefined/null, and BaseOffsetInMinutes is set as more than 0, Cisco TMS will throw an exception.
Standard		

TimeChange

Attribute	Read/Write	Description
ChangeSecondAtDay		The exact number of seconds from midnight that the change will occur. Must be between 0 and 86399.
TimeChangeAbsoluteRule	r/w, defining one of these two is mandatory	Used if DST changes happen at fixed dates. Specify the month and the day of the month. ■ Valid month values are between 1 and 12. ■ Valid day of the month values are between 1 and 31.
TimeChangeRelativeRule		Used if the DST change date is relative, such as the third sunday of a particular month. Specify the month, the week of the month, and the day of the week. ■ Valid month values are between 1 and 12 ■ Day of the week must be between 0 (Sunday) and 6 (Saturday). Default is 0 ■ Week of the month must be between 1 and 5. Default is 5, last week of the month.

If time zone validity rules are violated, the API will throw an exception, see Error Codes and Error Handling, page 52.

Participant

Attribute	Read/Write	Description
ParticipantId	r/w – optional	For Cisco TMS System Entities, this value must be the SystemId of the system. For external participants this value may be set, but is not required. If not set for external participants, Cisco TMS will create an ID with an integer greater than 0.
NameOrNumber	r/w – optional	For external participants, the participant name for dial-ins, or the fully qualified number to dial for dial-outs. For example: ▪ A dial-in can be given the value `Placeholder for John Doe`. ▪ An ISDN dial-out would be given the value `+1 (555) 1231234`. This value is required for external dial-out participants, and must be the fully qualified number to dial. Fully qualified numbers are of the format +CC (AC) BN where: ▪ CC=Country Code ▪ AC=Area Code ▪ BN=Base Number If the country does not use Area Codes, that element can be omitted completely, and the format would be +CC BN.

Attribute	Read/Write	Description
ParticipantCallType	r/w – required	The participant type. Valid values are: ■ *TMS*, a TMS System Entity. When this is specified, the **ParticipantId** must be the Cisco TMS System Entity ID as given in Cisco TMS. ■ *IP Video <-* or *ISDN Video <-*, an IP/ISDN video dial-in. If this is specified, you may give the participant a name using the **NameOrNumber** field. Cisco TMS will automatically give the participant an ID (less than 0). ■ *IP Tel <-* or *Telephone <-*, an IP/ISDN audio dial-in. If this is specified, you may give the participant a name using the **NameOrNumber** field. Cisco TMS will automatically give the participant an ID (less than 0). ■ *IP Video ->* or *ISDN Video ->*, an IP/ISDN video dial-out site. Specifying this value requires providing Cisco TMS with the number to use in the **NameOrNumber** field. (Formats: ISDN: `+1 (555) 1231234`, H.323 IP E.164: `12312321`, H323 IP Address: `10.0.0.10`). ■ *IP Tel ->* or *Telephone ->*, an IP/ISDN audio dial-out site. If this is specified, you must give Cisco TMS the number to use in the NameOrNumber field (Formats: ISDN: `+1 (555) 1231234`, H.323 IP E.164: `12312321`, H.323 IP Address: `10.0.0.10`). Call will be placed using 64kbps/54kbps depending on restrictions. ■ *Directory*, a Cisco TMS phone book entry. ■ *SIP ->* or *SIP <-*, a SIP video participant. The value can be a number or a URI. ■ *SIP Tel ->* or *SIP Tel <-*, a SIP audio participant. The value can be a number or a URI. ■ *TMS Master Participant*, this is the Video Conference Master, the participant that will be prompted to start the conference if the conference is set up as a manual connect, and to extend the conference just before it is due to end. When this entity is specified, the **ParticipantId** must be the Cisco TMS System Entity ID as given in Cisco TMS. It is only possible to specify a single Master Participant per conference, which must be a Cisco TMS System. Note that the system you specify as the TMS Master Participant must already exist in the participant list, with a ParticipantCallType of **TMS** and a matching **ParticipantId**. The *User* participant type is not currently supported by the API.

RecurrencePattern

Attribute	Read/Write	Description
FrequencyType	r/w required	The frequency of the recurrence rule. Legal values are: ■ *Daily* ■ *DailyWeekday* ■ *Weekly* ■ *Monthly* ■ *Default* *Default* is used by Cisco TMSXE to identify single telepresence meetings that are part of a non-telepresence series on the client side. Available, but unsupported values are: ■ *Secondly* ■ *Minutely* ■ *Hourly* ■ *Yearly*
Interval	r/w, required	Every X day/week/month as selected by **FrequencyType**
DaysOfWeek	r/w, optional	SOAP array of days of week in UTC equivalent. For example, a meeting that occurs after 5PM on Monday in UTC -7 will be on Tuesday in UTC. Include if relevant for your **FrequencyType**. Valid values are full names of days, capitalized. Include the name of each weekday that the meeting will occur.
FirstDayOfWeek	r/w, optional	First day of week. Used to split **DaysOfWeek** into "every X weeks" weekly patterns. The default value is *Sunday*.
BySetPosition	r/w, optional	Relative position of the instance in a pattern. The specific days must be defined in **DaysOfWeek**. For example, in a monthly pattern: ■ a value of 2 where **DaysOfWeek** is MONDAY, means the second Monday of every month. ■ -1 where *DaysOfWeek* is TUESDAY means the last Tuesday of every month. If set to *0* for monthly recurrence, **ByMonthDay** must be populated. If both are populated, an exception will be thrown.

Attribute	Read/Write	Description
ByMonthDay	r/w, optional	Absolute position of the instance in a pattern. For example, in a monthly pattern, a value of 2 means the second of the month. The valid range is 1-31. For months that have fewer days than the value specified, the last day of the month will be booked. If set to 0 for monthly recurrence, BySetPosition must be populated. If both are populated, an exception will be thrown.
PatternEndType	r/w, optional	End type: ▪ by number of occurrences ▪ by date (default) ▪ never (not currently supported)
PatternEndDateUTC	r/w, optional	In the case where **PatternEndType** is by date, this gives the end date of the recurrence pattern. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.
FirstOccurrenceRecInstanceIdUTC	r, optional	Gives the original start time of the first occurrence of this meeting. If the meeting is not an exception to the recurrence pattern, this time will be the same as the start time of the meeting. If the meeting time for the occurrence has been modified, this string gives the original start time according to the recurrence pattern. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.
PatternInstances	r/w, optional	In the case where **PatternEndType** is by number of occurrences, defines the number of occurrences to generate from the pattern.
Exceptions	r/w, optional	Exceptions to the pattern are supported using the **GetRecurrentConferenceById** and **SaveConferenceWithMode** functions. To get a conference with all its exceptions, use **GetRecurrentConferenceById**. To update a conference with exceptions, use the **SaveConferenceWithMode** function providing the exceptions in the **RecurrencePattern.Exceptions** array before saving the conference. As an alternative: 1. Use **GetConferenceIdByExternalId** with **RecInstanceIdUTC** (UTC string that points to the UTC day of the instance) to get conference id for the instance. 2. Use **SaveConferenceRecInstance** to save this exception.

ISDNBandwidth

Attribute	Read/Write	Description
Bandwidth	r/w, optional	The ISDN bandwidth that will be used when dialing the conference participants and when creating the conference. Note that *Max* is not currently supported. Example value `12b/768kbps`. If *Default* is selected, the value is set to the default conference ISDN bandwidth as defined in the **Conference Settings** page in Cisco TMS.

IPBandwidth

Attribute	Read/Write	Description
Bandwidth	r/w, optional	The IP bandwidth that will be used when dialing the conference participants and when creating the conference. Note that *Max* is not currently supported. Example value `12b/768kbps`. If *Default* is selected, the value is set to the default conference IP bandwidth as defined in the **Conference Settings** page in Cisco TMS.

Conference Object XML

The XML document below describes the Conference object.

```
<Conference>
	<ConferenceId>int</ConferenceId>
	<Title>string</Title>
	<StartTimeUTC>string</StartTimeUTC>
	<EndTimeUTC>string</EndTimeUTC>
	<RecurrenceInstanceIdUTC>string</RecurrenceInstanceIdUTC>
	<RecurrenceInstanceType>string</RecurrenceInstanceType>
	<FirstOccurrenceRecInstanceIdUTC>string</FirstOccurrenceRecInstanceIdUTC>
	<RecurrencePattern>
		<FrequencyType>Daily or DailyWeekday or Weekly or Monthly or Yearly or
		Secondly or Minutely or Hourly or Default</FrequencyType>
		<Interval>int</Interval>
		<DaysOfWeek>
		<DayOfWeek>Sunday or Monday or Tuesday or Wednesday or Thursday or Friday
		or Saturday</DayOfWeek>
		<DayOfWeek>Sunday or Monday or Tuesday or Wednesday or Thursday or Friday
		or Saturday</DayOfWeek>
		</DaysOfWeek>
		<FirstDayOfWeek>Sunday or Monday or Tuesday or Wednesday or Thursday or Friday
		or Saturday</FirstDayOfWeek>
		<BySetPosition>int</BySetPosition>
		<PatternEndType>EndByDate or EndByInstances or EndNever or Default</PatternEndType>
		<PatternEndDateUTC>string</PatternEndDateUTC>
		<FirstOccurrenceRecInstanceIdUTC>string</FirstOccurrenceRecInstanceIdUTC>
		<PatternInstances>int</PatternInstances>
		<Exceptions>
			<RecurrenceException  xsi:nil="true"  />
			<RecurrenceException  xsi:nil="true"  />
		</Exceptions>
	</RecurrencePattern>
	<OwnerId>long</OwnerId>
	<OwnerUserName>string</OwnerUserName>
	<OwnerFirstName>string</OwnerFirstName>
	<OwnerLastName>string</OwnerLastName>
	<OwnerEmailAddress>string</OwnerEmailAddress>
	<ConferenceType>Reservation Only or Automatic Call Launch or Manual Call Launch
	or Default or One Button To Push</ConferenceType>
	<Bandwidth>1b/64kbps or 2b/128kbps or 3b/192kbps or 4b/256kbps or 5b/320kbps
```

```
or 6b/384kbps or 8b/512kbps or 12b/768kbps or 18b/1152kbps or 23b/1472kbps
or 30b/1920kbps or 32b/2048kbps or 48b/3072kbps or 64b/4096kbps or Max or 6000kbps
Default</Bandwidth>
<PictureMode>Continuous Presence or Enhanced CP or Voice Switched
or Default</PictureMode>
<Encrypted>Yes or No or If Possible or Default</Encrypted>
<DataConference>Yes or No or If Possible or Default</DataConference>
<ShowExtendOption>Yes or No or Default or AutomaticBestEffort</ShowExtendOption>
<Password>string</Password>
<BillingCode>string</BillingCode>
<ISDNRestrict>boolean</ISDNRestrict>
<ConferenceInfoText>string</ConferenceInfoText>
<UserMessageText>string</UserMessageText>
<ExternalSourceId>string</ExternalSourceId>
<ExternalPrimaryKey>string</ExternalPrimaryKey>
<DetachedFromExternalSourceId>string</DetachedFromExternalSourceId>
<DetachedFromExternalPrimaryKey>string<DetachedFromExternalPrimaryKey>
<Participants>
        <Participant>
                <ParticipantId>int</ParticipantId>
                <NameOrNumber>string</NameOrNumber>
                <ParticipantCallType>TMS or IP Video <- or IP Tel <- or ISDN Video <-
                or Telephone <- or IP Video -> or IP Tel -> or ISDN Video ->
                or Telephone -> or Directory or User or SIP <- or SIP -> or SIP Tel <-
                or SIP Tel-> or 3G <- or 3G -> or TMS Master Participant
                </ParticipantCallType>
        </Participant>
        <Participant>
                <ParticipantId>int</ParticipantId>
                <NameOrNumber>string</NameOrNumber>
                <ParticipantCallType>TMS or IP Video <- or IP Tel <- or ISDN Video <-
                or Telephone <- or IP Video -> or IP Tel -> or ISDN Video ->
                or Telephone -> or Directory or User or SIP <- or SIP -> or SIP Tel <-
                or SIP Tel-> or 3G <- or 3G -> or TMS Master Participant
                </ParticipantCallType>
        </Participant>
</Participants>
<RecordedConferenceUri>string</RecordedConferenceUri>
<ExternalConference>
        <WebEx>
                <MeetingKey>string</MeetingKey>
                <SipUrl>string</SipUrl>
                <MeetingPassword>string</MeetingPassword>
                <JoinMeetingUrl>string</JoinMeetingUrl>
                <HostMeetingUrl>string</HostMeetingUrl>
                <HostKey>string</HostKey>
                <JoinBeforeHostTime>string</JoinBeforeHostTime>
                <Telephony xsi:nil="true" />
                <Errors xsi:nil="true" />
        </WebEx>
        <ExternallyHosted>
                <DialString>string</DialString>
        </ExternallyHosted>
        <WebExState>
                <WebExInstanceType>Normal or Delete or Modify</WebExInstanceType>
        </WebExState>
</ExternalConference>
<WebConferencePresenterUri>string</WebConferencePresenterUri>
<WebConferenceAttendeeUri>string</WebConferenceAttendeeUri>
<ISDNBandwidth>
        <Bandwidth>1b/64kbps or 2b/128kbps or 3b/192kbps or 4b/256kbps or
        5b/320kbps or 6b/384kbps or 8b/512kbps or 12b/768kbps or 18b/1152kbps
        or 23b/1472kbps or 30b/1920kbps or 32b/2048kbps or 48b/3072kbps
        or 64b/4096kbps or Max or 6000kbps or Default</Bandwidth>
```

```
        </ISDNBandwidth>
        <IPBandwidth>
                <Bandwidth>1b/64kbps or 2b/128kbps or 3b/192kbps or 4b/256kbps or
                5b/320kbps or 6b/384kbps or 8b/512kbps or 12b/768kbps or 18b/1152kbps
                or 23b/1472kbps or 30b/1920kbps or 32b/2048kbps or 48b/3072kbps
                or 64b/4096kbps or Max or 6000kbps or Default</Bandwidth>
        </IPBandwidth>
        <ConferenceLanguage>string</ConferenceLanguage>
        <ConferenceTimeZoneRules>
                <TimeZoneRule>
                        <ValidFromdateTime</ValidFrom
                        <Id>string</Id>
                        <BaseOffsetInMinutes>int</BaseOffsetInMinutes
                        <Daylight  xsi:nil="true" />
                        <DaylightOffsetInMinutes>int</DaylightOffsetInMinutes>
                        <Standard  xsi:nil="true" />
                </TimeZoneRule>
        </ConferenceTimeZoneRules>
        <ConferenceState>
                <Status>All or AllExceptDeleted or Pending or Ongoing or Finished or
                PendingAndOngoing or MeetingRequest or Rejected or NotSaved or Defective
                or Deleted</Status>
        </ConferenceState>
        <Version>int</Version>
</Conference>
```

Booking API functions

This reference section describes all the available functions of the booking API.

DeleteConferenceByExternalId

Delete a conference using a conference ID from an external source, usually Exchange.

This function is primarily intended for use by Cisco TMSXE.

Supported parameters:

`ExternalSourceId`	Unique identifier of the external source.
`ExternalConferenceId`	Unique identifier of the conference within the external source (primary key in database).

DeleteConferenceById

Deletes a conference with the given ConferenceId (as defined in Cisco TMS). If the conference is part of a recurring series, the whole series will be deleted.

Supported parameters:

`ConferenceId`	The ConferenceId of the conference to delete.

Returned data: Nothing.

DeleteConferenceInstanceByExternalId

Delete an occurrence of a series using a conference ID from an external source, usually Exchange.

This function is primarily intended for use by Cisco TMSXE.

Supported parameters:

`ExternalSourceId`	Unique identifier of the external source.

`ExternalConferenceId`	Unique identifier of the conference within the external source (primary key in database).
`RecurrenceIdUTC`	Identifies an instance in a series of conferences. A UTC-formatted datetime string. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.

DeleteConferenceRecInstanceById

Deletes an occurrence of a recurring conference with the given ConferenceId (as defined in Cisco TMS). This function is typically used when deleting a single meeting in a recurring series.

Supported parameters:

`ConferenceId`	The ConferenceId of the conference to delete.

Returned data: Nothing.

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

EndConferenceByExternalId

End an ongoing conference using an ID from an external source.

This function is primarily intended for use by Cisco TMSXE.
Supported parameters:

`ExternalSourceId`	Unique identifier of the external source.
`ExternalConferenceId`	Unique identifier of the conference within the external source (primary key in database).
`RecurrenceIdUTC`	Identifies an instance in a series of conferences. A UTC-formatted datetime string. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.

EndConferenceById

Ends an ongoing conference with the given **ConferenceId** (as defined in Cisco TMS). The conference will be set to *Finished*, and the end time will be set to the time of execution of the function. This function is typically used to end a running conference from a third party front-end GUI. The function will fail with an error if the conference has not yet started.

Supported parameters:

`ConferenceId`	The ConferenceId of the conference to delete.

Returned data: Nothing.

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetConferenceById

Get the available information about a particular conference.

Supported parameters:

`ConferenceId`	The ID of the conference (Based on Cisco TMS IDs)

Returned data: A Conference object based on the `ConferenceId`.

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetConferencesForUser

This function returns all conferences that were created by or that are owned by a particular user, for a specified period of time.

Supported parameters:

`UserName`	The Cisco TMS user to get bookings for. If no user name is provided (empty string), the logged in user is used.
`StartTime`	The start and end time of bookings. The time is given in UTC format.
`EndTime`	
`ConferenceStatus`	An enumeration of what type of conferences will be fetched from Cisco TMS. The available types are: ■ *All* ■ *AllExceptDeleted* ■ *Pending* ■ *Ongoing* ■ *Finished* ■ *PendingAndOngoing* ■ *MeetingRequest* (Conference has been requested by a user without booking rights, needs approval.) ■ *Rejected* ■ *NotSaved* ■ *Defective* ■ *Deleted*

Returned data: An array with Conference objects.

The conference objects will not include:

- the list of participants
- WebEx conference information
- recording URIs
- ConferenceInfoHtml or ConferenceInfoText

GetConferenceBookingEventMail

Used to retrieve content for email notifications on booking events; not invites, typically errors or warnings.

Supported parameters:

`Conference`	The conference object on which to base the email message.

`Message`	A message from the client to inject into the email notification. Contains: ■ A MessageType which determines the color and prefix of the message box at the top of the email message. Supported values are *Information* (green), *Warning* (yellow), and *Error* (red). ■ One or more lines of text strings containing the actual message.
`ContentTypes`	Determines whether the email notification will be sent as *PlainText* or *Html*.
`Language`	A list of supported languages can be gotten from the Remote Setup API. See GetConferenceLanguages, page 21.

Returned data: A list of conference booking event e-mail content. Setting Language to blank will return the conference language.

GetConferenceByExternalId

Get the conference with the given ExternalSourceId and ExternalConferenceId.

This function is primarily intended for use by Cisco TMSXE.

Supported parameters:

`ExternalSourceId`	Unique identifier of the external source.
`ExternalConferenceId`	Unique identifier of the conference within the external source (primary key in database).

Returned data: A Conference object. If the conference is part of a recurrent series, existing exceptions to the series are returned in the RecurrencePattern Exceptions array.

GetConferenceIdByExternalId

This function is used to look up a conference that has been updated in the external source, and that must be updated in Cisco TMS. The `ExternalSourceId` and `ExternalPrimaryKey` fields must have been provided with the initial performance of the SaveConferenceWithMode function.

This function is typically used when information about a conference reserved in the external application is needed. The GetConferenceById function is used to get information about the conference from Cisco TMS.

Supported parameters:

`ExternalSourceId`	Unique identifier of the external source.
`ExternalConferenceId`	Unique identifier of the conference within the external source (primary key in database).
`RecurrenceIdUTC`	Identifies an instance in a series of conferences. A UTC-formatted datetime string. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.

Returned data: A ConferenceId, as defined in Cisco TMS.

GetConferenceInstanceByExternalId

Get an occurrence from a series with the given ExternalSourceId, ExternalConferenceId, and RecurrenceIdUTC.

This function is primarily intended for use by Cisco TMSXE.

Supported parameters:

`ExternalSourceId`	Unique identifier of the external source.

`ExternalConferenceId`	Unique identifier of the conference within the external source (primary key in database).
`RecurrenceIdUTC`	Identifies an instance in a series of conferences. A UTC-formatted datetime string. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.

Returned data: A Conference object.

GetConferenceInviteMail

Used to retrieve content for conference invite email notifications.

Supported parameters:

`ExternalSourceId`	Unique identifier of the external source.
`ExternalConferenceId`	Unique identifier of the conference within the external source (primary key in database).
`RecurrenceIdUTC`	Identifies an instance in a series of conferences. A UTC-formatted datetime string. Only UTC date-time groups ending with Z are supported. Example: 1975-06-01 23:32:11Z.
`Messages`	A list of messages from the client to inject into the email notification. Contains: ■ A MessageType which determines the color and prefix of the message box at the top of the email message. Supported values are *Information* (green), *Warning* (yellow), and *Error* (red). ■ One or more lines of text strings containing the actual message.
`ContentTypes`	An array of content types. If multiple types are included, both *PlainText* and *Html* will be used (sent as Multipart).
`Language`	A list of supported languages can be retrieved from the Remote Setup API. See GetConferenceLanguages, page 21.

Returned data: A list of conference invite e-mail content. Setting Language to blank will return the conference language.

GetConferencesForSystems

This function returns all conferences for a list of systems between two dates. It is typically used to build a display of resource availability information in external application for a specific system when the external application does not store its own resource availability information.

The function should be used with caution. If a large number of conferences are booked between the two dates in Cisco TMS, it will take a long time to process the result of this function.

Supported parameters:

`SystemIds`	An array of IDs of the systems, based on Cisco TMS IDs.
`StartTime`	The start and end time of bookings. The time is given in UTC format.
`EndTime`	

`ConferenceStatus`	An enumeration of the type of conferences that will be fetched from Cisco TMS. The available types are: ■ *All* ■ *AllExceptDeleted* ■ *Pending* ■ *Ongoing* ■ *Finished* ■ *PendingAndOngoing* ■ *MeetingRequest* (Conference has been requested by a user without booking rights, needs approval.) ■ *Rejected* ■ *NotSaved* ■ *Defective* ■ *Deleted*

Returned data: An array with Conference objects. As of API version 11, this includes scheduled conferences only, no ad hoc conferences.

The conference objects will not include:

- the list of participants
- WebEx conference information
- recording URIs
- ConferenceInfoHtml or ConferenceInfoText

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetDefaultConference

Creates a default conference object with ID equals -1 based on the conference settings specified in Cisco TMS.

This function is typically used as a basis for new meetings, where all that is needed is to define the start and end time, along with the participants in the conference.

Supported parameters: None

Returned data: A Conference object using the default values defined in Cisco TMS.

- The start time of the conference is set to the current time.
- The end time is set to the start time + **Default Scheduled Call Duration (in minutes)** as configured in **Administrative Tools > Configuration > Conference Settings**.
- If, in **Administrative Tools > Configuration > WebEx Settings**, the setting for **Add WebEx To All Conferences** is *Yes*, the default conference will always include WebEx, whether or not this is exposed to the user by the client. For more information about including WebEx, see ExternalConference, page 31.

GetRecordingAliases

Supported parameters:

`UserName`	The user to retrieve recording alias for. If no UserName is provided (empty string), the logged in user will be used.

Returned data: An array of RecordingDevice, where the key is the string representation of a recording device name, or a recording cluster name.

The value is an array of AliasInfo for that particular recording device/cluster, holding an AliasId (string) and a SystemId (int) . The AliasId and SystemId can be used to add a recording participant to a conference.

GetRecurrentConferenceById

Returns a Conference object with the given ConferenceId. If the conference is a recurrent conference, existing exceptions to the recurrent series are returned in the RecurrencePattern Exceptions array of the returned Conference object.

Supported parameters:

`ConferenceId`	The ID of the conference, based on TMS IDs.

Returned data: A Conference object based on the ConferenceId.

If the recurrent series has been deleted, the returned object will include a negative conference ID.

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetRecurrentConferenceByIdWithFirstOngoingOrPendingStartTime

This function is primarily intended for use by Cisco TMSPE Smart Scheduler.

Supported parameters:

`ConferenceId`	The ID of the conference, based on TMS IDs.

Returned data: A Conference object containing, ff the conference is a recurrent series with exceptions, aRecurrencePattern Exceptions array.

The start time of the conference will be mapped to the first ongoing or pending conference occurrence.

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetTimeZoneRulesById

Get the available time zone rules for the given time zone.

Supported parameters:

`idString`	A string containing the ID of the time zone on the Cisco TMS server.

Returned data: An array of TimeZoneRule elements

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetTransactionsSince

This function is used to get a list of conference creations, updated and deletions that must be performed in order to keep a mirrored conference database synchronized. Note that:

- If the number of matching conferences exceeds the internal limit, the array's **TransType** element will be *Incomplete*. To get all transactions, the client must send the last returned ID of the array as the **CurrentTransactionId** to receive the next set, until the **TransType** is no longer returned as *Incomplete*.
- The transaction identified as **CurrentTransactionId** will not be included in the returned array.

Supported parameters:

`CurrentTransactionId`	The transaction ID of the last committed transaction of the last synchronization.

Returned data: An array of transactions since the **CurrentTransactionId**. This array will also contain the **TransType** element, which may be *New* or *Updated* or *Deleted* or *ListIncomplete*.

If the provided ID does not exist, this will cause an error. See the section Error Codes and Error Handling, page 52.

GetTransactionsSinceWithExternalId

Get a list of conference create, update and delete operations that must be performed in order to keep a mirrored conference database synchronized.

This function is primarily intended for use by Cisco TMSXE.

See GetTransactionsSince , page 49.

SaveConferenceWithMode

This function saves a conference in Cisco TMS.

- If an **ExternalPrimaryKey** is provided, Cisco TMS will try to load an existing conference using the **ExternalSourceId** and **ExternalPrimaryKey**.
- If **ExternalPrimaryKey** is not provided or the conference is not found, Cisco TMS will check the **ConferenceId**.
  - If **ConferenceId** is < 0, Cisco TMS creates and saves a new conference, provided that the conference can be booked.
  - If **ConferenceId** is > 0, Cisco TMS looks for an existing conference and updates it, or throws a "meeting not found" error. See the section Error Codes and Error Handling, page 52.

Depending on the selected booking mode, the function will fail or cause a defective conference in the following scenarios:

- One or more of the participants are already booked in the same time period.
- A call route is needed, but no call route could be found.

If this function is performed on a recurrent series, the entire series is affected.

See Booking Modes, page 5 for more detail on modes and defective conferences.

Supported parameters:

`Conference`	The Conference object to be created/updated.
`BookingMode`	*BestEffortForced* or *Strict*.

Returned data: A Conference object updated with actual values saved in Cisco TMS.

SaveConference

This legacy function works identically to SaveConferenceWithMode, page 50, but automatically uses the *Strict* booking mode.

SaveConferenceRecInstanceWithMode

Used to update a single occurrence of a recurrent conference series in Cisco TMS. Similar to SaveConferenceWithMode, page 50.

Supported parameters:

`Conference`	The Conference object to be created/updated
BookingMode	*BestEffortForced* or *Strict*.

Returned data: A Conference object updated with actual values saved in Cisco TMS.

SaveConferenceRecInstance

This legacy function works identically to SaveConferenceRecInstanceWithMode, page 50, but automatically uses the *Strict* booking mode.

SaveConferences

Saves a list of conferences to Cisco TMS, with the option to save either all or none depending on availability information.

Use this function if the recurrence pattern of the Conference object does not support the recurrence model in the external application.

Supported parameters:

Conference	An array of conference objects.
oneTransaction	*True* if the objects should be booked as one transaction, meaning that either all or none of the meetings will be booked depending on the free/busy information. Currently only *true* is supported for this function.

Returned data: An array of Conference objects updated with actual values saved in Cisco TMS.

Error Codes and Error Handling

This chapter provides an overview of the available error codes as well as an error handling example.

Error Codes 52
Error Handling 54

Error Codes

Note that the description for each error code describes the typical scenario, but not necessarily the exact error message. Messages will vary based on the specific error situation.

Error	Code	Description
LICENSE	-2147219503	Client attempts to book more systems than there are licenses for.
DATABASE_DOWN	-2147219500	There is a problem with the Cisco TMS database.
MEETINGNOTFOUND	-2147218302	Client tries to access (get or update) a conference that does not exist.
SYSTEMNOTFOUND	-2147218301	Client tries to access (get or update) a system/participant that does not exist in Cisco TMS.
SYSTEM_ALREADY_BOOKED	-2147218300	Client tries to schedule a participant that has already been scheduled.
SYSTEM_NOT_ALLOWED_ IN_BOOKING	-2147218262	Client tries to book a system that the user does not have booking rights for, or the system has **Allow Bookings** set to *False*.
MEETINGNOTACTIVE	-2147218272	Client tries to end a conference that is not active.
MEETINGISDELETED	-2147218271	Client tries to end a conference that has been deleted.
MEETINGACTIVE	-2147218270	Client tries to modify the start time of a conference that is already active.
CANNOTBOOKINTHEPAST	-2147218269	Client tries to book a conference with an end time that is in the past.
NO_ACCESS_TO_ CONFERENCE	-2147218266	Client tries to get a conference, but the user does not have the permissions in Cisco TMS to read all conferences.
START_TIME_AFTER_MAX_ NUMBER_OF_DAYS_IN_ FUTURE	-2147218265	Client tries to save a outside of the booking window set in Cisco TMS.
NEVER_ENDING_ RECURRENCE_NOT_ SUPPORTED	-2147218264	Client tries to save a conference with a "NeverEnding" recurrence end type, which is not supported.

Error	Code	Description
START_TIME_AFTER_END_TIME	-2147218263	Client tries to save a conference where the start time is later than the end time.
INVALID_TIMEZONE_INFO	-2147218268	The specified time zone information is not valid.
WEBEX_SITE_NOT_FOUND	-2147218267	WebEx site not found. (The invoked WebEx site URL does not exist.)
WEBEX_ERROR	-2147218260	Any other error regarding WebEx will use this code.
SECURITY	-2147218259	Client tries to modify or delete a conference, but the user does not have the required permissions in Cisco TMS.
SERVER_BUSY	-2147218258	Cisco TMS is currently unable to handle the client's request. There are two possible causes: ■ A database operation has timed out. This is more likely to occur in a large deployment. ■ There are too many concurrent SaveConferenceWithMode and/or DeleteConference requests. When receiving this message, repeat the request at a later time.
UNKNOWN	-2147218261	Any exception not covered by other error codes.
CANNOT_ADD_PARTICIPANT	-2147218256	Client attempted to add dial-in participants to a *Reservation Only* conference.
UNSUPPORTED_RECURRENCE	-2147218257	The client tried to book an unsupported recurrence pattern. Scenarios include: ■ Invalid number of occurrences ■ More than one occurrence on the same day ■ Attempt to move an occurrence to start later than the subsequent occurrence.
INVALID_DATETIME_FORMAT	-2147218254	The date was provided in the wrong (non-UTC) format.
DIAL_OUT_NUMBER_NOT_SET	-2147218251	One or more dial-out participants do not have a dial out number configured
CONCURRENT_MODIFICATION	-2147218248	The client tried to modify a booking, but the booking changed before the modification was submitted.
NOT_ENOUGH_MCU_RESOURCES	-2147218250	Client attempted to book a meeting for which the necessary bridge resources are not available.
PARTICIPANT_ERROR	-2147218249	There is a problem with one or more participants that the client tried to book. Scenarios include: ■ Missing participants ■ Necessary protocols not enabled for participants ■ Configuration error on participant (system)

Error	Code	Description
INSTANCE_NOT_FOUND_ WITHIN_PATTERN	-2147218252	The client tried to save or get a conference occurrence that is not part of the recurrence pattern of the series. Error message "No such occurrence".
ERR_CLIENT_SESSION_ BLANK	-2147218247	The client provided a session ID that was blank or null, which is not permitted. Cisco TMS will add a new session ID on the thrown SOAP exception which the client can use in subsequent calls.
ERR_CLIENT_SESSION_ EXPIRED	-2147218253	The provided client session ID has expired. Cisco TMS will add a new session ID on the thrown SOAP exception which the client can use in subsequent calls.
ERR_USER_NOT_FOUND	-2147218255	The provided user information could not be resolved to an existing user, and Cisco TMS was not able to create a new user. See the server logs for details.

If an exception is thrown, the exception message will contain the reason. If you get "Unspecified Exception" / "Unspecified Error", this usually means that there is a syntax flaw in the conference sent to the SaveConferenceWithMode function.

In such a case, an error description would be given in the Cisco TMS log files, which you can download from the Cisco TMS website, or locate in the folder (**/tms/data/logs/tmsdebug/log-web.txt**) on the Cisco TMS server.

Error Handling

If the Cisco TMS server is operational with the proper licenses, errors are caused by sending the wrong parameters to the API, such as trying to create a booking in the past, or trying to get systems, users, or conferences from Cisco TMS using the wrong ID. When an exception is caught, it is generally an indication that the client call must be changed before it is sent again.

Exceptions

All errors generated from the API are SoapExceptions, hence each time a save operation is performed against the API, the code should handle exceptions of type SoapException.

The message field of the exception will contain a string with a description of what went wrong. In many cases, displaying this information to the user will be helpful.

HTTP Error 401

The server will normally return the HTTP error code 500 Internal Server Error for the SoapExceptions.

If the HTTP error code 401 Unauthorized is received, the user credentials supplied were not authorized to access the server.

Code Examples

This chapter provides examples of how to apply the remote setup API and the booking API in your development.

Code examples are provided in C#.

Remote Setup API Examples 55
Booking API Setup 56
Using Different Participant Types 56
Externally Hosted Conference Example 57
Recording Participant Example 57
Booking Example Where One Occurrence is Saved as Defective 58
Saving and Updating Recurrent Conferences 59
Time Zone Handling 60
Error Handling Examples 61

Remote Setup API Examples

The code snippet below demonstrates how to authenticate with the remote setup API.

```
public void InitRemoteSetupService()
{
    // Specify username and password to authenticate to service.
    // (Can also be done in web.config)
    var credentials = new NetworkCredential(Username, Password, Server);

    remoteSetupService = new RemoteSetupService { Credentials = credentials };
    if (remoteSetupService.ExternalAPIVersionSoapHeaderValue == null)
    {
        remoteSetupService.ExternalAPIVersionSoapHeaderValue = new
RemoteSetupServiceWS.ExternalAPIVersionSoapHeader();
    }
    remoteSetupService.ExternalAPIVersionSoapHeaderValue.ClientVersionIn = 18;
}
```

The code snippet below demonstrates how to loop through all systems in Cisco TMS, and display information about each system.

```
public void DisplaySystemInformation()
{
    InitRemoteSetupService();
    // Get all systems from TMS
    var tmsSystems = remoteSetupService.GetSystems();
    // Loop through the systems and output information about each system
    foreach (RemoteSetupServiceWS.TMSSystem tmsSystem in tmsSystems)
    {
        Console.Out.WriteLine("SystemId: " + tmsSystem + " System Name:" + tmsSystem.SystemName);
    }
}
```

Booking API Setup

The code snippet below demonstrates how to authenticate with Cisco TMSBA and specify the version.

```
public void InitBookingService()
{
    // Specify username and password to authenticate to service.
    // (Can also be done in web.config)
    var credentials = new NetworkCredential(Username, Password, Server);

    bookingService = new BookingService();
       bookingService.Credentials = credentials;
    if (bookingService.ExternalAPIVersionSoapHeaderValue == null)
    {
        bookingService.ExternalAPIVersionSoapHeaderValue = new BookingServiceWS.ExternalAPIVersionSoapHeader
();
    }
    bookingService.ExternalAPIVersionSoapHeaderValue.ClientVersionIn = 18;
    bookingService.ExternalAPIVersionSoapHeaderValue.ClientSession = "clientSessionString"
}
```

Using Different Participant Types

When using the API as a web reference, the participant types, such as "IP Video", "ISDN Video", and so on are created as enumerations called, for example, `IPTel`, `IPTel1`. Values ending in 1 are dial-out, values not ending in 1 are dial-ins.

The code snippet below shows how to create a conference with three different participant types. An MCU is required for this call to be saved.

```
public void SaveConferenceWithVariousParticipantTypes()
{
    InitBookingService();
    // Get a default conference object from TMS, where most common values are set
    // (using default values specified in TMS)
    var conference = bookingService.GetDefaultConference();

    // Create and initialize an array of conference participants, and add it to the conference
    conference.Participants = new[]
        {
            new Participant
                {
                    // Adds a SIP Dial-in participant
                    NameOrNumber = "SIP Dial-in 1", ParticipantCallType = ParticipantType.SIP
                },
            new Participant
                {
                    // Adds a SIP Dial-out participant
                    NameOrNumber = "manager1@example.com", ParticipantCallType = ParticipantType.SIP1
                },
            new Participant
                {
                    // Adds a TMS participant (endpoint)
                    ParticipantCallType = ParticipantType.TMS, ParticipantId = 4
                }
        };

    // Save the conference, saving the returned conference (where all values are now specified)
    conference = bookingService.SaveConference(conference);

    PrintConferenceInformation(conference);
}
```

Externally Hosted Conference Example

The code snippet below demonstrates how to create an Externally Hosted Conference, that is, a conference with a given SIP video address.

```
public void SaveConferenceWithExternallyHostedConference()
    {
        InitBookingService();
        var conference = bookingService.GetDefaultConference();

        // Add two SIP dial in participants, and one meeting room
        conference.Participants = new[]
            {
                new Participant
                   {
                        NameOrNumber = "SIP Dial-in 1", ParticipantCallType = ParticipantType.SIP
                   },
                new Participant
                   {
                        NameOrNumber = "SIP Dial-in 2", ParticipantCallType = ParticipantType.SIP
                   },
                new Participant
                   {
                        // Adds a TMS participant
                        ParticipantCallType = ParticipantType.TMS, ParticipantId = 4
                   }
            };

        // Add the externally hosted conference
        conference.ExternalConference = new ExternalConference
            {
                ExternallyHosted = new ExternalHost
                   {
                        DialString = "externalhost@example.com",
                   }
            };

        // Save the conference, saving the returned conference (where all values are now specified)
        conference = bookingService.SaveConference(conference);

        PrintConferenceInformation(conference);
    }
```

Recording Participant Example

The code snippet below demonstrates how to create a conference with two participants. One of the participants is a recording participant, the other a video system registered in TMS.

```
public void SaveConferenceWithRecordingParticipant()
{
    InitBookingService();
    var conference = bookingService.GetDefaultConference();

    // Create the elements of the array (the actual participants)
    // Create one dial-out participant
    var dialOutParticipant = new Participant {ParticipantCallType = ParticipantType.IPVideo1, NameOrNumber =
"10.0.1.2"};
    // Get the recording aliases for the logged in user
    var recordingDevicesWithAliases = bookingService.GetRecordingAliases("");
    var recordingParticipant = new Participant();
    bool foundAliasInformation = false;
    if (recordingDevicesWithAliases != null && recordingDevicesWithAliases.Any())
    {
```

```
        // use the first recording device in the arrayvar recordingAlias = recordingDevicesWithAliases.First
();
        if (recordingAlias.Aliases != null && recordingAlias.Aliases.Any())
        {
            foundAliasInformation = true;
            // use the first alias found on the first recording device
            AliasInfo aliasInfo = recordingAlias.Aliases.First();
            recordingParticipant.ParticipantCallType = ParticipantType.TMS;
            recordingParticipant.ParticipantId = aliasInfo.SystemId;
            recordingParticipant.NameOrNumber = aliasInfo.AliasId;
        }
    }

    if (foundAliasInformation)
    {
        conference.Participants = new []
            {
                dialOutParticipant,
                recordingParticipant
            };
    }
    else
    {
        // no alias information found in TMS
        conference.Participants = new[]
            {
                dialOutParticipant
            };
    }

    // Save the conference, saving the returned conference (where all values are now specified)
    conference = bookingService.SaveConference(conference);

    PrintConferenceInformation(conference);
}
```

Booking Example Where One Occurrence is Saved as Defective

The code snippet below demonstrates how to create a conference with two participants. One of the participants is a recording participant, the other a video system registered in TMS.

```
public void SaveConferenceOneInstanceBecomesDefective()
    {
        InitBookingService();

        var start = DateTime.Now.AddHours(1);
        var end = start.AddMinutes(10);

        var conflictingConference = bookingService.GetDefaultConference();

        // Conflicting conference starts tomorrow
        conflictingConference.StartTimeUTC = start.AddDays(1).ToString("u");
        conflictingConference.EndTimeUTC = end.AddDays(1).ToString("u");

        var conflictingParticipant = new Participant
            {
                ParticipantCallType = ParticipantType.TMS,
                ParticipantId = 1009
            };

        // Add two SIP dial in participants, and one meeting room
        conflictingConference.Participants = new[] { conflictingParticipant };

        // Save the conference, saving the returned conference (where all values are now specified)
```

```
        conflictingConference = bookingService.SaveConference(conflictingConference);


        // Create a daily, recurrent conference, which will conflict on instance number 2 (tomorrow)var
conference = bookingService.GetDefaultConference();
        conference.StartTimeUTC = start.ToString("u");
        conference.EndTimeUTC = end.ToString("u");

        conference.Participants = new[] { conflictingParticipant };

        conference.RecurrencePattern = new RecurrencePattern
            {
                FrequencyType = RecurringFrequency.Daily,
                PatternEndType = RecurrenceEndType.EndByInstances,
                PatternInstances = 3,
                Interval = 1,
            };

        // This save invocation will result in the 2nd instance in the recurrent series being defective

        // The returned conference will be marked with ConferenceState == Defective if > 0 instances are
defective in the series:

        var savedConferenceResult = bookingService.SaveConferenceWithMode(conference,
BookingMode.BestEffortForced);
        var savedConference = bookingService.GetRecurrentConferenceById
(savedConferenceResult.Conference.ConferenceId);
        Assert.That(savedConference.ConferenceState.Status, Is.EqualTo(ConferenceStatus.Defective));

        PrintConferenceInformation(conflictingConference);
    }
```

Saving and Updating Recurrent Conferences

The code snippet below demonstrates how to save/update a conference series.

```
public void SaveAndUpdateRecurrentConference()
{
    InitBookingService();
    var conference = bookingService.GetDefaultConference();

    // Set the conference to start in the future (default is now)
    var start = DateTime.Now.AddHours(1);
    var end = start.AddMinutes(10);

    conference.StartTimeUTC = start.ToString("u");
    conference.EndTimeUTC = end.ToString("u");

    conference.Participants = new[]
        {
            new Participant{ParticipantCallType = ParticipantType.SIP, NameOrNumber = "Sip dial-in 1"},
            new Participant{ParticipantCallType = ParticipantType.SIP, NameOrNumber = "Sip dial-in 2"}
        };

    // setup the recurrence pattern
    conference.RecurrencePattern = new RecurrencePattern
        {
            FrequencyType = RecurringFrequency.Daily,
            PatternEndType = RecurrenceEndType.EndByInstances,
            PatternInstances = 10,
        };
    // Save the conference, saving the returned conference (where all values are now specified)
    conference = bookingService.SaveConference(conference);

    // update the conference, change the pattern from Daily to Weekly and change from 10 instances to 5
```

```
    conference.RecurrencePattern = new RecurrencePattern
    {
        FrequencyType = RecurringFrequency.Weekly,
        PatternEndType = RecurrenceEndType.EndByInstances,
        PatternInstances = 5,
    };

    conference = bookingService.SaveConference(conference);

    PrintConferenceInformation(conference);
}
```

Time Zone Handling

The code snippets in this section demonstrate how to save a regular and a custom set of time zone rules.

Regular Time Zone

```
public void SaveConferenceWithRegularTimeZone()
{
    InitBookingService();
    var conference = bookingService.GetDefaultConference();
    var start = DateTime.Now.AddHours(1);
    var end = start.AddMinutes(10);

    conference.StartTimeUTC = start.ToString("u");
    conference.EndTimeUTC = end.ToString("u");

    conference.RecurrencePattern = new RecurrencePattern
    {
        FrequencyType = RecurringFrequency.Daily,
        Interval = 1,
        PatternEndType = RecurrenceEndType.EndByInstances,
        PatternInstances = 3,
    };

    try
    {
        var timeZoneRules = bookingService.GetTimeZoneRulesById("Central Standard Time");
        if (timeZoneRules != null && timeZoneRules.Length > 0)
        {
            conference.ConferenceTimeZoneRules = timeZoneRules;
            conference = bookingService.SaveConference(conference);

            PrintConferenceInformation(conference);
        }
    }
    catch (Exception)
    {
        Console.WriteLine("TMS did not return any time zone information for given time zone id");

    }
}
```

Custom Time Zone

```
public void SaveConferenceWithCustomTimeZone()
{
    var conference = bookingService.GetDefaultConference();
    var start = DateTime.Now.AddHours(1);
    var end = start.AddMinutes(10);

    conference.StartTimeUTC = start.ToString("u");
```

```
    conference.EndTimeUTC = end.ToString("u");

    conference.RecurrencePattern = new RecurrencePattern
    {
        FrequencyType = RecurringFrequency.Daily,
        Interval = 1,
        PatternEndType = RecurrenceEndType.EndByInstances,
        PatternInstances = 10,
    };

    conference.ConferenceTimeZoneRules = new[]
    {
        new TimeZoneRule
           {
                Id = "My custom rule",
                BaseOffsetInMinutes = 60,
                DaylightOffsetInMinutes = 60,
                ValidFrom = DateTime.MinValue,
                Daylight = new TimeChange
                {
                    ChangeSecondAtDay = 2*60*60, // Daylight changes at 02:00 am
                    AbsoluteRule = new TimeChangeAbsoluteRule
                    {
                        Month = 3,
                        DayOfMonth = 3,
                    }
                },
                Standard = new TimeChange
                {
                    ChangeSecondAtDay = 10*60*60, //// Daylight changes at 10:00 am
                    RelativeRule = new TimeChangeRelativeRule
                    {
                        Month = 10,
                        DayOfTheWeek = 0, // Sunday
                        WeekOfTheMonthIndex = 5, // Last
                    }
                }
            }
    };
    conference = bookingService.SaveConference(conference);

    PrintConferenceInformation(conference);
}
```

Error Handling Examples

The following code examples demonstrate how to handle errors generated from API calls.

Conference in the Past

Running this code will output the message: "You cannot book a conference in the past".

```
public void ErrorHandling_BookInThePast()
{
    InitBookingService();
    var conference = bookingService.GetDefaultConference();
    var start = DateTime.Now.AddHours(-10);
    var end = start.AddMinutes(10);

    conference.StartTimeUTC = start.ToString("u");
    conference.EndTimeUTC = end.ToString("u");

    try
    {
```

```
        conference = bookingService.SaveConference(conference);
    }
    catch (SoapException e)
    {
        Console.WriteLine("Got error with error code {0}, and message {1}, from Booking API",
e.Detail.InnerXml, e.Message);
    }

    PrintConferenceInformation(conference);
}
```

System Unavailable

Running the code below will output a "system unavailable" message.

```
public void ErrorHandling_SystemNotAvailable()
{
    InitBookingService();
    var conference = bookingService.GetDefaultConference();
    var start = DateTime.Now.AddHours(1);
    var end = start.AddMinutes(10);

    conference.StartTimeUTC = start.ToString("u");
    conference.EndTimeUTC = end.ToString("u");

    var participant = new Participant {ParticipantCallType = ParticipantType.TMS, ParticipantId = 4};

    conference.Participants = new []
        {
            participant
        };
    bookingService.SaveConference(conference);

    try
    {
        // By setting the Id to -1, we try to book a new conference, with the same time and participant
        conference.ConferenceId = -1;
        conference = bookingService.SaveConference(conference);
    }
    catch (SoapException e)
    {
        Console.WriteLine("Got error with error code {0}, and message {1}, from Booking API",
e.Detail.InnerXml, e.Message);
    }

    PrintConferenceInformation(conference);
}

private static void PrintConferenceInformation(Conference conference)
{
    // Output information about the conference.
    Console.Out.WriteLine(conference.ConferenceInfoText);
    Console.Out.WriteLine(conference.UserMessageText);
    Console.Out.WriteLine(conference.ConferenceId);
}

public void Dispose()
{
    bookingService.Dispose();
    remoteSetupService.Dispose();
}
```

Cisco Legal Information

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

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

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

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

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

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

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

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

© 2016 Cisco Systems, Inc. All rights reserved.

Cisco Trademark

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