

**TANDBERG**  
**TANDBERG**  
**Management Suite**  
**3rd Party Booking**  
**API**  
**Version 5.0**

**V 11.9 or later of TMS**

The TANDBERG Management Suite (TMS) 3rd Party Booking API is an API that gives developers access to the booking functionality in TMS. The interface is used by TANDBERG in its Microsoft Exchange and IBM Lotus Domino implementations, and provides the same feature set as the user interface for the TANDBERG Scheduler.

### **Intended audience**

The target audience for this document are developers that are required to implement a data/audio/video conferencing solution towards a booking interface not supported by TMS directly or where TMS does not provide the needed functionality, for example a company specific Enterprise Resource Planning system (ERP). Such booking systems will be referred to as external booking systems in the document.

D13566 Rev.09

This document is not to be reproduced in whole or in part without permission in writing from:

**TANDBERG**

## TABLE OF CONTENTS

**Functional overview 3**

**Document structure 3**

**Basic Entities 4**

**Booking principles 5**

**Usage patterns 6**

**API Reference 8**

**Code examples 23**

## Functional overview

There are four main usages of the API, which are often used in conjunction with one-another:

1. **Importing:** Importing and/or displaying resources from TMS in an external application, using the remote setup API. By using this API, the process of importing systems in TMS into the third party application can be automated, or done through a GUI.
2. **Synchronizing:** Synchronizing resources booked in TMS with resources available in an external booking system. Using this part of the API allows an external system to keep track of booking transaction on the TMS server, and to synchronize itself with booking done through TMS. (Direction: TMS -> External Booking System).
3. **Booking:** Booking of resources booked in an external booking system in TMS. Using this part of the API allows you to forward booking requests from an external booking system to TMS, and reserving the resources there. (Direction: External Booking System -> TMS)
4. **Availability:** Read bookings in TMS. Using this part of the API lets you display the TMS reservations, stored in the internal reservation database of TMS.

The API requires that the master database for resource bookings is the TMS database. Therefore, it is not possible to forward booking requests made in TMS to an external booking system.

**The API cannot be used for system management, call management or other features beyond booking.**

## Document structure

This document is divided into the following parts:

- Basic entities - describes the objects and entities of the 3rd Party Booking API.
- Booking principles - presents and overview of how to use the 3rd Party Booking API.
- Usage patterns - describes how an external GUI can take advantage of the booking engine in TMS and the way an external booking system, with its own booking database can interoperate with TMS and take advantage of the booking engine. The Microsoft Exchange and IBM Lotus Domino integrations work against the API in this manner.
- API Reference – describes the functions and objects available in the 3rd Party Booking API and the remote setup API.

### TMS System Entity

A TMS System Entity is an entity used to describe an item that can be booked. In the TMS user interface, TMS system entities are seen as systems and rooms (e.g. the entities viewable in the System Navigator). Neither phone book entries or web conference servers are systems.

In TMS each systems has a unique identifier or id. This id is visible in the user interface as of TMS 9.5. TMS allows a single system to be located in multiple folders, however the underling system entity (and id) will be equal for all instances of the system in TMS.

The table in the database that contains systems is the objSystem table. It is not recommended to update this table manually; however reading information should not cause issues.

### TMS Conference Entity

A TMS Conference Entity is an entity that describes a reservation in TMS (Conferences in TMS are also known as Bookings). All conferences in TMS must make at least one reservation of a TMS System Entity. E.g. it is not possible to create a conference that contains only phone book entries. TMS will at the time the conference is saved add the required MCU reservations (the TMS System Entity) to allow the call to complete.

One limitation is when reserving web conference resources. Due to Web Conference resources not being TMS System Entities, it is not possible to only reserve a web conference; you will also need to include at least one TMS system entity reservation.

A conference is stored in the database in the ScheduledCall table. Each conference has a unique identified (id). The TMS System Entity reservation, dial-in slots, phone book entries etc. are stored in the ScheduledParticipant table. This table is coupled to the ScheduledCall table with the foreign key the ScheduledCall.Id from ScheduledParticipant.ScheduledCallId.

A conference id can be seen in the TMS user interface under Booking -> All Meetings -> Id column.

### TMS User Entity

A TMS User Entity is an entity that holds information about users in TMS. All persons who books conferences in TMS will be part of the TMS users. In addition to the name, TMS holds information about time zone and e-mail for all users.

## Booking principles

By using the third party booking API, users can book video resources from their own booking application.

Systems can be read from TMS using the remote service API of the Third Party Booking API to import and/or show them in the booking application.

One type of API usage is to make a front-end GUI utilizing the booking capabilities of TMS. By importing systems using the remote service API, the booking application can show availability information about systems in TMS. This lets users book conferences through the application using the booking API.

Another approach is when you have a separate booking system with its own reservation database. When a system is booked from this application, it contacts TMS to find out if the system is available by trying to book the system in TMS through the third party booking API. If this succeeds, the system can be booked in the third party application as well.

In some cases, the third party application will use a service to contact TMS that runs under a service account, so the user's credentials will not be used for booking the meeting. In cases like this, it's important that the service account has needed privileges to book a meeting on behalf of others, if user ownership of the meeting should be sent to TMS. The remote setup API includes a method for creating such an account, and also verification mechanisms to make sure the required credentials are present during creation of the service account.

It's only possible to list and book endpoints and rooms from the API, as TMS will allocate the needed network resources for you. E.g. if a user books 5 endpoints, this is sent to TMS. TMS will then determine if network resources like a MCU are required, and automatically reserve these resources.

To make sure that meetings booked in TMS are exported to the third party application, synchronization is used to update the database of the third party application from the TMS database. The synchronization mechanism should be run fairly often (e.g. every 10th minute) to pick up the latest meetings booked from TMS, and importing them into the third party application. To reduce the delay of meeting imports from TMS to the third party application, the service account could be configured to receive e-mail every time a meeting is created/deleted or changed from TMS. When an e-mail is received, the third party application can then start the synchronization method, hence reducing the time it will take for the meeting to be imported.

### GUI Pattern

This part describes how the APIs are used to build a front-end GUI for a booking application. It describes three main components:

1. **Displaying:** Importing resources from TMS in an external application. By using the remote setup API, systems in TMS can be exported to a front-end GUI and used to display system entities in TMS.
2. **Availability:** Read bookings in TMS to display availability information about TMS entities. Using this part of the API lets you make display the TMS reservations, using TMS's internal reservation database. The users in TMS can be read, to do filtering on bookings on users in the external front-end GUI.
3. **Booking:** Booking resource booked in an external booking system in TMS. Using this part of the API allows you to forward booking requests from an external booking system to TMS, and reserving the resources there. (Direction: External Booking System -> TMS)

#### Displaying

Use the *GetSystems* or *GetSystemsForUser* function to get a list of available systems in TMS. This function returns a list of *TMSSystem* objects that includes the needed information, like the Id of the system, to show in the front-end GUI. *GetSystems* will return all systems in TMS, while *GetSystemsForUser* will only return the systems the user has booking privileges for. If control of system access is done in the GUI application, use *GetSystems* and filter the systems in the application.

#### Availability

Use the *GetConferences* to get all reservations between two dates in from TMS. *GetConferencesForSystem* is used to get availability information about a particular system. This information can be used to display E.g. a calendar showing availability information in an external application.

If filtering of availability information is required, use information in the Conference object. *GetUsers* returns all users registered in TMS. The output of this function can be used to display a drop-down list of all users in TMS, with the ability to only show conferences that are booked by a specific person.

#### Booking

Use the *GetDefaultConference* functions to get Conference objects with TMS defined default values for Conference properties.

Use *GetConferenceById* or *GetConferenceIdByExternalId* functions to retrieve already saved conferences.

To save changes to a conference, edit the properties on the Conference and use function *SaveConference*. This will save the conference to TMS if the validation of the properties is OK, if not an exception will be raised.

To delete a conference use the *DeleteConferenceById* function. Conference participants will be disconnected if the conference is deleted while it is active or connected.

### Synchronization pattern

This part describes how the APIs are used in conjunction external booking applications with their own reservation database. It describes three main components:

1. **Importing:** Importing resources from TMS in an external application. By using this API, the process of importing systems in TMS into the third party application can be automated, or done through a GUI.
2. **Synchronizing:** Synchronizing resources booked in TMS with resources available in an external booking system. Using this part of the API allows an external system to keep track of booking transaction on the TMS server, and to synchronize itself with booking done through TMS. (Direction: TMS -> External Booking System). This part is not applicable if a GUI front end is being developed, that does not have its own reservation database.
3. **Booking:** Booking resource booked in an external booking system in TMS. Using this part of the API allows you to forward booking requests from an external booking system to TMS, and reserving the resources there. (Direction: External Booking System -> TMS)
4. **Availability:** Read bookings in TMS. Using this part of the API lets you make display the TMS reservations, using TMS's internal reservation database.

## Importing

Use the *GetSystems* or *GetSystemsForUser* function to get a list of available systems in TMS. This function returns a list of TMSSystem objects that includes the needed information, like the Id of the system, to import to a third party application. *GetSystems* will return all systems in TMS, while *GetSystemsForUser* will only return the systems the user has booking privileges for. If control of system access is done in the external application, use *GetSystems* and filter the systems in the application.

## Synchronizing

Use the *GetTransactionsSince* function to get a list of transactions since a given transaction Id (all Conferences have a transaction Id property). The list of transaction contains the transaction type (New, Update, and Delete) and an associated ConferenceId, use the *GetConferenceById* to get an updated Conference object – and update the conference within the external source. The current transaction Id should then be updated to the last Conference's TransactionId.

## Booking

Use the *GetDefaultConference* functions to get Conference objects with TMS defined default values for Conference properties.

Use *GetConferenceById* or *GetConferenceIdByExternalId* functions to retrieve already saved conferences.

To save changes to a conference, edit the properties on the Conference and use function *SaveConference*. This will save the conference to TMS if the validation of the properties is OK, if not an exception will be raised.

To delete a conference use the *DeleteConferenceById* function. Conference participants will be disconnected if the conference is deleted while it is active or connected.

### API Reference

The TMS 3rd Party Booking API provides a two Web Services API towards the booking engine of TMS. Web Services allows for simple integration into most common languages and programming environments. See your development tool reference for information on how to build implementation stubs to help speed the development application that use Web Services.

The WSDL file for the TMS Remote setup API is located at:

<http://127.0.0.1/tms/external/booking/remotesetup/remotesetupservice.asmx>

The WSDL file for the TMS 3rd Party Booking API is located at:

<http://127.0.0.1/tms/external/Booking/BookingService.asmx>

If you are using Microsoft Visual Studio .Net, the simplest way to reference the APIs is to select Project -> Add Web Reference, then enter the URLs above (exchanging 127.0.0.1 with the name of the web-server TMS is installed on). If you are using network load balancing, it is recommended to use the clusters virtual IP-address or DNS-name for this task, this will then also allow fail over for the API.

To use the TMS 3rd Party Booking API you will need one Application Integration License per server using the API. Please contact TANDBERG for more information (<http://www.tandberg.com>)

To import from TMS or book meetings through the API, you must first authenticate yourself to the APIs. To be able to book meetings using the API, you need to at least have Misc Booking rights.

**On a default TMS installation, the APIs requires the use of Windows Challenge Response or NTLM authentication. Not all environments support this authentication mechanism (non Windows based environments), so you may need to allow for Basic Authentication on the /TMS/external/booking virtual directory (this can be done using the Internet Information Services manager). It is not recommended to allow anonymous authentication – but if you choose to do so, the IUSR\_<machinename> needs to be given Book on behalf of permissions in TMS.**

### API version

The third party booking API has gone through several versions. It is backwards compatible, meaning that applications that are written for an older version of the API will still work. However, to utilize the full potential of the API, the API version needs to be specified in the headers when the functions of the API are called. The current API version is version 4, so this is the number that needs to be set to utilize the latest version of the API. Do not set a number greater than 4 as this is likely to break compatibilities when using later revisions of the API.

### ExternalAPIVersionSoapHeader

Each call made to the 3rd party booking API should include a 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.

```

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

```

## Remote setup API Reference

### TMSSystem Object

The TMSSystem object contains information about a system in TMS. This object is used to read information from TMS; remote setup API does not support updating system information in TMS.

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

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

```

<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 Un-
known</SystemStatus>
  </SystemStatus>
</TMSSystem>

```

Note that all fields are not required, so the output might contain less information about each system than the object potentially can hold.

#### TMSSystem:

- **SystemId** – The Id of the system in TMS. Use this to refer to the associated system in TMS from your application. E.g. when booking a conference, insert the ids of the chosen systems into the Conference object.
- **SystemName** – The name of the system in TMS. Use this to display the name of the system in your application.
- **Contact** – The system contact associated with the system in TMS.
- **Manufacturer** – The manufacturer of the system. E.g. TANDBERG
- **Description** – A textual description stored in TMS. This file can contain information like number of chairs in the meeting room the system is located.
- **SystemType** – The type of system, e.g. TANDBERG 1000MXP
- **NetworkAddress** – The IP or DNS address of the system.
- **Location** – The ISDN location where the system is located.
- **ISDNNumber** – The ISDN number of the system.
- **QNumber** – The fully qualified ISDN number of the system. A fully qualified ISDN number always includes the country code and area code.
- **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 offset.

#### SystemCategory:

- **SystemCategory** - An enumeration value of what category of a system this is. E.g. endpoint.

#### 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.

### TMS User object

The TMS user object contains information about TMS users. Use this object to access information about users in 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>
  <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>

```

#### User:

- **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 if 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.

## Functions/Methods

### DisableConferenceAPIUser

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

**This function is typically used during uninstall procedures.**

#### Input:

**userName** - the full user name in NT4 style (domain\username) of the user to delete.

### GenerateConferenceAPIUser

This function generates a TMS Booking API account in the default user container of the TMS server, including registering the user in TMS (as a hidden user not in normal user lists). The user is added to the Site Administrator Group. The user is configured to receive e-mail scheduling event notifications for all creation/update/deletions of bookings. This method is used during installation to create a separate user for the booking API.

The current user must be a TMS Site Administrator, along with being a local computer Administrator in order for the method to complete. The e-mail scheduling event notifications are typically used for updating the external booking system with changes done on the TMS server.

**This function is typically used during install/setup procedures.**

#### Input:

**userNameBase** - The base portion of the user name. If a user with the name already exists a numeric postfix is added (e.g. tms-confuser ==> tms-confuser1).

**encPassword** - A base64 encoded password that is to be used for the newly

created user.

**emailAddress** - The email address of the user.

**sendNotifications** - If the user shall receive scheduling notifications.

Returns the user name of the created user (NT4 domain/username style).

### **GetSystemById**

This function returns information about a specific system. If the system is not found an exception is thrown.

**Input:**

**TMSSystemId** - system Id as given in TMS.

Returns a TMSSystem object.

### **GetSystems**

This function returns all endpoints and rooms registered in TMS. Note that no network systems, like MCUs are returned, since this is normally not booked by the users, but is being added to the conference by TMS if required.

**Typically used during set-up of resources in the external booking system to connect resources in TMS with resources in the external booking system.**

**Input:**

None

Returns an array of TMSSystem objects.

### **GetSystemsForUser**

This function returns all endpoints and rooms that can be booked by the current user, using the credential for the account used to communicate with the third party booking API of TMS. Note that no network systems, like MCUs are returned, since this is normally not booked by the users, but is being added to the conference by TMS if required.

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

**Input:**

None

Returns an array of TMSSystem objects.

### **IsAlive**

This is used to check the connection to the web-services of TMS.

**Typically used during installation to check the URL to this web-service.**

**Input:**

None

Returns a boolean value True/False. True if the connection works.

## GetUsers

This function returns all users registered in TMS.

**This function is typically used in front-end GUIs to provide a list of TMS users, and give the ability to filter output from the third party booking API based on users from this output.**

### Input:

None

Returns an array of User objects.

## IsLocalAdmin

This function checks if the current user can create local / Active Directory accounts in the default user container on the TMS server.

**This is typically used during installation to check if the user installing the integration has sufficient access towards Active Directory. This method should return True in order for the GenerateConferenceAPIUser method to succeed.**

### Input:

None

Returns a boolean value True/False. True if the user is a local admin user.

## IsTMSServiceUser

This function is used to check if the current user is flagged as an Exchange Integration user and has access to book on behalf of other users.

**This is typically used during installation to check if the user installing the integration has enough access towards the TMS server.**

### Input:

None

Returns a boolean value True/False. True if user is a TMS service user.

## IsTMSSiteAdmin

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

**This is typically used during installation to check if the user installing the integration has enough access towards the TMS server. This method should return True in order for the GenerateConferenceAPIUser method to succeed.**

### Input:

None

Returns a boolean value True/False. True if the user is a TMS Site Administrator.

# Booking API Reference

## Conference Object

Use this object to read and write conference properties like Start Time, End Time, Conference Title, Conference Password etc. Also, use this object for conference call related values like Bandwidth, Picture mode, Encryption mode etc.

All conference resources (video participants, audio participants, phone book participant, external participants etc.) are also held in this object, together with the call route for connecting the resources.

You also define the conference type in Conference;

- Automatic call launch, which will connect the added participant at conference start time and disconnect them again at conference end time.
- Manual call launch, which ask the conference master participant to connect the call at conference start time.
- Reservation Only, which only reserves the added participants for the conference duration.

Conference data can be saved/updated, and handled by TMS using the Save-Conference function described below.

The XML document below describes the Conference object. Following the XML is a description of the elements and what format input values require.

```
<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 Ad-Hoc conference</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 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</Data-
Conference>
  <ShowExtendOption> Yes or No or Default</ShowExtendOption>
  <Password>string</Password>
  <BillingCode>string</BillingCode>
  <ISDNRestrict>boolean</ISDNRestrict>
  <ConferenceInfoText>string</ConferenceInfoText>
  <UserMessageText>string</UserMessageText>
  <ExternalSourceId>string</ExternalSourceId>
  <ExternalPrimaryKey>string</ExternalPrimaryKey>
  <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</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</ParticipantCallType>
    </Participant>
  </Participants>
  <RecordedConferenceUri>string</RecordedConferenceUri>
  <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 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 Default</Bandwidth>
  </IPBandwidth>
</Conference>

```

## Conference:

- **ConferenceId** (r/w - optional, if not specified -1 is assumed) – initially set to -1 to state to the SaveConference method that the conference needs to be created. If set to a value greater than 0, the existing conference with the given Id is replaced with the conference specified.
- **Title** (r/w - optional - If none specified the default name as defined in the Administrator Tools Page in TMS is used
- **StartTimeUTC** (r/w - required) - the start time of the conference in UTC in the Universal sortable date/time pattern, “yyyy-MM-dd HH:mm:ss’Z”  
Example: 2010-06-01 22:32:11Z.
- **EndTimeUTC** (r/w - required) - the end time of the conference in UTC in the Universal sortable date/time pattern, “yyyy-MM-dd HH:mm:ss’Z”  
Example: 2010-06-01 23:32:11Z.
- **RecurrenceInstancelIdUTC** (r - only used when getting conference from TMS) - Gives the start date 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.
- **RecurrenceInstanceType** (r - only used when getting conference from TMS) - If this string contains the value ‘ modify’ it means that the particular meeting is an exception to a recurrence pattern. If the string contains ‘deleted’, it is a meeting that has been deleted from a series of recurring meetings.
- **FirstOccurrenceReclInstancelIdUTC** (r - only used when getting conference from TMS) - Gives the start date of first instance of the meeting according to the recurrence pattern. If this is different from StartTimeUTC, the first meeting of the series is an exception to the recurrence pattern.
- **RecurrencePattern** (r/w – optional) – sets the recurrence patterns for recurrent meetings. This is valid if you call the ‘SaveConferenceReclInstance’ method.
- **OwnerId** (r/w - optional) – the user Id of the owner of the conference. This is typically not set by external booking APIs, but by TMS. If no owner is specified the user authenticated to the WebService is used. IDs of users existing in TMS can be found in the aclUser table of the TMS database.
- **OwnerUserName** (w - optional) – the user name of the person booking the conference. This is used by TMS to look-up the OwnerId in the TMS database. If no owner is specified the user authenticated to the WebService is used. If OwnerId is specified, it will be used instead of this field.
- **OwnerFirstName/OwnerLastName/OwnerEmailAddress** (w - optional) – the first and last name of the owner of the conference. This is used by TMS to look-up the OwnerId in the TMS database. If no owner is specified the user authenticated to the WebService is used. If OwnerUserName or OwnerId is used instead of this field, those fields will be used instead of this field.
- **ConferenceType** (r/w - optional, if not specified, Default is assumed) – can be set to one of the values
  - Reservation Only – TMS reserves the resources, but will not set-up the call.
  - Automatic Call Launch – TMS will reserve the resources, and at the start time of the conference, connect the participant.
  - Manual Call Launch – TMS will reserve the resource, and wait for the VC Master (first TAA system in the Participant List) to select ‘Connect’. These calls can also be connected using the TMS Conference Control Center interface.
  - Default – Use the conference type/reservation type that is defined in the Administrator Tools Page in TMS as the default type.
- **Bandwidth** (Discontinued) – This item is for backwards compatibility, and

is no longer to be used. Use ISDNBandwidth and IPBandwidth instead to control the conference bandwidth.

- **PictureMode** (r/w - optional – if not specified, Default is assumed) – The picture mode/conference layout to use for the conference. Valid values are: Continuous Presence, Enhanced CP, Voice Switched and Default. If Default selected the default conference picture mode as defined in the Administrator Tools Page in TMS is used.
- **Encrypted** (r/w - optional – if not specified, Default is assumed) – The encryption mode for the conference. Valid values are: Yes, No, If Possible and Default. If Default is selected, the default encryption mode as defined in the Administrator Tools Page in TMS is assumed.
- **DataConference** (r/w - optional – if not specified, No is assumed) – If data conference should be added to the conference. Valid values are: Yes, No and If Possible.
- **ShowExtendOption** (r/w - optional, if not specified, Default is assumed) – Set this value to allow the VC Master (the first TAA endpoint in the participant list) is to get a message to extend the conference, when the conference is close to ending. If Default is specified, the default Show Extend Option defined in the Administrator Tools Page in TMS is used.
- **Password** (r/w - optional, if not specified, TMS may set a password if TMS is set to automatically generate passwords for new conferences). The password for the conference participants must enter to actually join the call itself.
- **BillingCode** (r/w - optional, if not specified, blank is assumed). The billing code to use for the conference. If TMS requires billing codes, this field must be specified and will be validated against the list of billing codes in TMS. If no match is found, the conference will not be created.
- **ISDNRestrict** (r/w - option, if not specified, No is assumed) – If the ISDN channels should be restricted (e.g. use 54k and not 64k)
- **ConferenceInfoText** (r – only used when getting conference from TMS) – Information on how the conference is to connect. Call Route, etc.
- **UserMessageText** (r/w – optional – blank if not specified) – A user definable text/description of the conference.
- **ExternalSourceId/ExternalPrimaryKey** (r/w – optional – blank if not specified) – A user definable external source and id, this is used to synchronize the TMS database with the external sources database. If TMS is given a value for these fields, TMS will return the value for all instances of the same conference.
- **Participants** – (r/w, required) List of conference participants. When calling GetDefaultConference, the participant list will be empty.
- **RecordedConferenceUri** (r – only used when getting conference from TMS) If the conference is recorded, this is the Uri of the conference recording.
- **WebConferencePresenterUri** (r – only used when getting conference from TMS) If the conference includes a web conference, this is the Uri of the presenters web conference.
- **WebConferenceAttendeeUri** (r – only used when getting conference from TMS) If the conference includes a web conference, this is the Uri of the attendee web conference. If this field and WebConferencePresenterUri is the same, they will both include the same information
- **ISDNBandwidth** (r/w - optional – if not specified, Default is assumed) The ISDN bandwidth of the conference
- **IPBandwidth** (r/w - optional – if not specified, Default is assumed) The IP bandwidth of the conference

#### Participant:

- **ParticipantId** – (r/w – optional) – For 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, TMS will create a Id with an integer less 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. E.g. dial-in can be given the value ‘Placeholder for John Doe’, whereas 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=AreaCode, BN=Basenumber. If the country does not use Area Codes, that element can be omitted completely and the format would be +CC BN.
- **ParticipantCallType** – (r/w – required) – The participant type. Valid values are
  - TMS – A TMS System Entity. When this is specified, the ParticipantId must be the TMS System Entity Id as given in 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. TMS will automatically give the participant and 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. TMS will automatically give the participant and Id (less than 0)
  - IP Video -> or ISDN Video -> – An IP/ISDN video dial-out site. If this is specified, you must give TMS the number to use in the NameOrNumber field (Formats: ISDN: +1 (555) 1231234, H323 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 TMS the number to use in the NameOrNumber field (Formats: ISDN: +1 (555) 1231234, H323 IP E.164: 12312321, H323 IP Address: 10.0.0.10). Call will be placed using 64kbps/54kbps depending on restrict.
  - User – Not used/supported

#### **RecurrencePattern:**

- **FrequencyType** – (r/w required) The frequency of the recurrence rule. Legal values are: Daily, DailyWeekly, Weekly, Monthly, Yearly, Secondly, Minutely, Hourly, Default.
- **Interval** – (r/w required) - Every X day/week/month as selected by FrequencyType
- **DaysOfWeek** - Days of week if FrequencyType is Weekly or the X mon/tue/ weekend/day etc.
  - For every X mon-sun, only set the day.
  - For every day, set all days.
  - For every weekday, set mon-fri
  - For every weekend day, set sat & sun
- **FirstDayOfWeek** - First day of week. Used to split DaysOfWeek[] in “every X week” weekly patterns. Default is Sunday.
- **BySetPosition** - Relative position of the instance in a pattern. For example, in a monthly pattern, a value of 2 means second day of month, -1 means last day of month. The allowed days must be defined in DaysOfWeek. 0 means monthly day of the meeting (every X day of a month).
- **PatternEndType** - End type: by number of occurrences, by date, or never (not supported)
- **PatternEndDateUTC** - In the case where PatternEndType is by date, this gives the end date of the recurrence pattern.
- **FirstOccurrenceReclInstanceUTC** - Gives the original start time of the

meeting of this occurrence. E.g. if the occurrence is to start at 12am a day, and this particular instance of the recurring meeting is an exception (that the meeting time has been moved), this string gives the original start time of the meeting according to the recurrence pattern. If the meeting is not an exception to the recurrence pattern, this time will be the same as the start time of the meeting.

- **PatternInstances** - In the case where PatternEndType is by number of instances, defines the number of instances to generate from the pattern.
- **Exceptions** - Exceptions to the pattern. Not currently supported. As an alternative, use GetConferenceIdByExternalId with RecInstanceIdUTC (UTC string that points to the UTC day of the instance) to get conference id for the instance, and use SaveConferenceRecInstance to save this exception.

#### **ISDNBandwidth:**

- **Bandwidth:** The ISDN bandwidth will be used when dialling the conference participants, and also used when creating the conference. Note Max is not supported at this time. Example value "3b/193kbps". If Default is selected, the value is set to the default conference ISDN bandwidth as defined in the Administrator Tools Page in TMS.

#### **IPBandwidth:**

- **Bandwidth:** The IP bandwidth will be used when dialling the conference participants, and also used when creating the conference. Note Max is not supported at this time. Example value "3b/193kbps". If Default is selected, the value is set to the default conference IP bandwidth as defined in the Administrator Tools Page in TMS.

## **Functions/Methods**

### **DeleteConferenceById**

Deletes a conference with the given ConferenceId (as defined in TMS). If the conference does not exist, an exception is thrown. If the conference is part of a recurring series, the whole series will be deleted.

#### **Input:**

**ConferenceId** - the ConferenceId of the conference to delete.

Returns nothing

### **DeleteConferenceRecInstanceById**

Deletes an occurrence of a recurring conference with the given ConferenceId (as defined in TMS). If the conference does not exist, an exception is thrown.

**This function is typically used when deleting a single meeting in a recurring series.**

#### **Input:**

**ConferenceId** - the ConferenceId of the conference to delete.

Returns nothing

## EndConferenceById

Ends a conference with the given Conferenceld (as defined in TMS). The conference will be set to finished, and the end time will be set to the time of execution of the method. If the conference is deleted or has not started yet, an exception is thrown.

### Input:

**Conferenceld** - the Conferenceld of the conference to delete.

**This function is typically used to end a running conference from a third party front-end GUI.**

Returns nothing

## GetConferenceById

This function gets information about a particular conference. If the conference does not exist, an exception is thrown.

### Input:

**Conferenceld** - The Id of the conference (Based on TMS Ids)

Returns a Conference object based on the Conferenceld.

## GetConferenceldByExternalId

Returns a Conferenceld (as defined in TMS) given an ExternalSourceId and ExternalConferenceld. This function is used to look up conference that have been updated in the external source, and that must be updated in TMS. The ExternalSourceId and the ExternalPrimaryKey fields must have been provided in the initial SaveConference call.

**This function is typically used when information about a conference reserved in the external application is needed. First this function is called to get the corresponding conference in TMS. The GetConferenceById is used to get information about the conference from TMS.**

### Input:

**ExternalSourceId** - Unique identifier of the external source (i.e. server IP-address).

**ExternalConferenceld** - Unique identifier of the conference within the external source (e.g. primary key in database).

Returns a Conferenceld, as defined in TMS.

## GetConferencesForSystem

This function returns all conferences for a list of systems between two dates.

**This function is typically used to build a display of free/busy information in external application for a specific system when the external application does not store its own free/busy information.**

This function should be used with caution. If lots of conferences are booked between the two dates in TMS, it will take long to process the result of this method.

**Input:**

SystemIds - An array of Ids of the systems (Based on TMS Ids)

StartDate - The start date of bookings

EndDate - The end date of bookings

ConferenceStatus - An enumeration of what type of conferences that will be fetched from TMS. (All, AllExceptDeleted, Pending, Ongoing, Finished, PendingAndOngoingm MeetingRequest, Rejected, Finished or Deleted)

Returns an array with Conference objects.

**GetDefaultConference**

Creates a default conference object based on the conference settings specified in 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.**

**Input:**

None

Returns a Conference object using the default values defined in TMS. The start time of the conference is set to the current time.

**GetTransactionsSince**

Returns an Array of Transactions since the CurrentTransactionId.

**This method is used to get a list of conference creations, updated and deletes that must be performed in order to keep a mirrored conference database synchronized. The transaction with id CurrentTransactionId will not be included in the array.**

**Input:**

**CurrentTransactionId** - The transaction id of the last committed transaction of the last synchronization.

Returns an Array of Transaction, giving the changes done since CurrentTransactionId

**SaveConference**

Saves a conference in TMS. If conferenceId is not set, a new conference is created and saved. If the conferenceId is set, the existing conference is updated. If no conference with the given ConferenceId exists, an exception is thrown.

This method will fail if any of the participants are already booked in the same time period or if a call route is to be made, but no call route could be found.

If this method is performed on a recurring conference, the complete series is affected.

**Input:**

**Conference** - the Conference object to be created/updated

Returns a Conference object updated with actual values saved in TMS.

If an exception is thrown, you will be given a reason in the exception message. If you get an Unspecified Exception/Unspecified Error, this usually means that there is a syntax flaw in the conference sent to the SaveConference function. In such a case, an error description would be given in the TMS-log files (<http://<tms-server address>/tms/data/logs/tmsdebug/log-web.txt> as of TMS9.5 or <c:\tmsdebug\log-web.txt> for older versions)

### SaveConferenceReclInstance

Saves an instance of a recurring conference in TMS. Similar to SaveConference except that this is used to modify an occurrence of a series of recurring conferences.

This function is typically used when updating a single instance of a series of recurring conferences.

#### Input:

**Conference** - the Conference object to be created/updated

Returns a Conference object updated with actual values saved in TMS.

### SaveConferences

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

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

#### Input:

**Conference[]** – An array of conference objects.

**oneTransaction** – True if they 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 method.

Returns as array of Conference objects updated with actual values saved in TMS.

### Visual Studio .Net using C# and Web-References

To use the 3rd Party Booking API in Visual Studio .Net, you need to add a Web Reference to your project (Project -> Add Web Reference), specify the URL to your TMS server: <http://127.0.0.1/tms/external/Booking/BookingService.asmx> for the booking API and <http://localhost/tms/external/booking/remotesetup/remotesetupservice.asmx> for the remote setup API. You will be required to authenticate against the web-services to create the reference.

#### Remote Setup API example

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

```
// Specify username and password to authenticate to service.
// (Can also be done in web.config)
NetworkCredential credentials = new NetworkCredential("xxx", "yyy",
"ZZZ");

RemoteSetupService remoteSetupService = new RemoteSetupService();
remoteSetupService.Credentials = credentials;

// Set API to use version 5
if (remoteSetupService.ExternalAPIVersionSoapHeaderValue == null)
    remoteSetupService.ExternalAPIVersionSoapHeaderValue = new Remote-
SetupService.ExternalAPIVersionSoapHeader();
remoteSetupService.ExternalAPIVersionSoapHeaderValue.ClientVersionIn = 5;

// Get all systems from TMS
TMSSystem[] tmsSystems = remoteSetupService.GetSystems();

// Loop through the systems and output information about each system
foreach (TMSSystem tmsSystem in tmsSystems)
{
    Console.Out.WriteLine("SystemId: " + tmsSystem.SystemId + " System
Name:" + tmsSystem.SystemName);
}
}
```

#### Booking API example

Note: When using the API as a web-reference, the ParticipantsTypes for "IP Video <-", "ISDN Video ->" etc are created as enumerations called IPTel, IPTel1, etc. The values with an ending 1 are the dial-out, whereas without the ending 1 are dial-ins.

The code snippet below show how to create a conference to two external partici- pants (specified by IP-address). An MCU is required for this call to be saved.

```
// Specify username and password to authenticate to service.
// (Can also be done in web.config)
NetworkCredential credentials = new NetworkCredential("xxx", "yyy",
"ZZZ");

BookingService bookingService = new BookingService();
bookingService.Credentials = credentials;

// Set API to use version 5
if (bookingService.ExternalAPIVersionSoapHeaderValue == null)
    bookingService.ExternalAPIVersionSoapHeaderValue = new BookingServ-
```

```

ice.ExternalAPIVersionSoapHeader();
bookingService.ExternalAPIVersionSoapHeaderValue.ClientVersionIn = 5;

// Get a default conference object, where most common values are set
// (using default values specified in TMS)
Conference conference = bookingService.GetDefaultConference();

// Create an array of participants
Participant[] participants = new Participant[2];

// Create the elements of the array (the actual participants)
participants[0] = new Participant();
participants[0].ParticipantCallType = ParticipantType.IPVideol; //
Dial-out video
participants[0].NameOrNumber = "10.47.8.170";

participants[1] = new Participant();
participants[1].ParticipantCallType = ParticipantType.IPVideol; //
Dial-out video
participants[1].NameOrNumber = "10.47.8.171";

// Add the participants to the conference.
conference.Participants = participants;

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

// Output information about the conference.
Console.Out.WriteLine(conference.ConferenceInfoText);
Console.Out.WriteLine(conference.UserMessageText);
Console.Out.WriteLine(conference.ConferenceId);

```

### Error handling example

The code show how to handle errors generated from API calls. If the TMS server is operational with the proper licenses, the errors are caused by sending wrong parameters to the API, like doing bookings in the past, or trying to get systems in TMS using the wrong ID.

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.

When an exception is caught, it is generally an indication that the client call must be changed before it is resent.

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

```

// Specify username and password to authenticate to service.
// (Can also be done in web.config)
NetworkCredential credentials = new NetworkCredential("xxx", "yyy",
"zzz");

BookingService bookingService = new BookingService();
bookingService.Credentials = credentials;

// Set API to use version 5

```

```

if (bookingService.ExternalAPIVersionSoapHeaderValue == null)
    bookingService.ExternalAPIVersionSoapHeaderValue = new Booking-
Service.ExternalAPIVersionSoapHeader();
bookingService.ExternalAPIVersionSoapHeaderValue.ClientVersionIn = 5;

// Get a default conference object, where most common values are set
// (using default values specified in TMS)
Conference conference = bookingService.GetDefaultConference();

// Create an array of participants
Participant[] participants = new Participant[2];

// Create the elements of the array (the actual participants)
participants[0] = new Participant();
participants[0].ParticipantCallType = ParticipantType.IPVideol; //
Dial-out video
participants[0].NameOrNumber = "10.47.8.170";

participants[1] = new Participant();
participants[1].ParticipantCallType = ParticipantType.IPVideol; //
Dial-out video
participants[1].NameOrNumber = "10.47.8.171";

// Add the participants to the conference.
conference.Participants = participants;

// Set start date to the 12th of December, 2000
DateTime startTime = new DateTime(2000, 12, 12, 10,00,00);
DateTime endTime = new DateTime(2000, 12, 12, 11,00,00);

conference.StartTimeUTC = startTime.ToUniversalTime().ToString("u");
conference.EndTimeUTC = endTime.ToUniversalTime().ToString("u");

// Save the conference, saving the returned conference (where all
values are now specified)
try
{
    conference = bookingService.SaveConference(conference);
}
catch (SoapException e)
{
    Console.WriteLine(e.Message);
}

```

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

**The server will return a HTTP error with error code 500 for the SoapExceptions. If the HTTP error code 401 is received, this is an authorization/authentication error, and means that the user's credentials supplied is not authorized to access the server.**

### Java date example

When working with Java, use the C# code example as guide. The date string can be formatted according to this example:

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss'Z'");
```