WYDE Web Services REST API – Programmer's Guide

Web Services version 1.2.1
Document Revision: 1.0.34
(August 11, 2015)

Table of Contents

Tables List

Figures List

Chapter 1: Introduction

WYDE Web Services SOAP API is obsolete (deprecated); it may be removed at some version in the future. Instead of SOAP services the new WYDE REST[1] API is created. This documents notes differences between old SOAP and new REST web services for the WYDE bridge software.

Open Data Protocol (OData)[2] protocol is used for creating and consuming data WYDE REST API. This protocol is used to define web methods calls, parameters, etc.

For WYDE bridge SOAP web services documentation see "*Web Services API – Programmer's Guide*" from the WYDE support Web site: http://docs.wydevoice.com/.

[1] See details about Representational State Transfer (REST) protocol at http://en.wikipedia.org/wiki/Representational_state_transfer
[2] See details at http://www.odata.org/docs/ and http://en.wikipedia.org/wiki/Open_Data_Protocol

Chapter 2: Common Conventions

Resource Naming Conventions

Each WYDE REST web service resource has its URL. Any information the server can provide is exposed as a resource.

There are several top level resources:
/subscribers
/callflows
/dids
/didGroups
/meetings
/meetingRooms
/calls
/mdrs
/cdrs
/callEvents
/events
/settings
/bridge
/bridges

Examples:
GET *https://<Wyde bridge domain>/subscribers* - returns all subscriber objects
GET *https://<Wyde bridge domain>/subscribers/247* - returns the subscriber with identifier equals to *247*

There are some "second level" resources that can be retrieved as:
/subscribers/247/meetingRooms
/subscribers/247/meetingRooms/283848/keychain
/meetings/283848/attributes

Examples:
GET *https://<Wyde bridge domain>/subscribers/247/meetingRooms* - returns all meeting rooms that assigned to the subscriber with identifier equals to *247*
GET *https://<Wyde bridge domain>/subscribers/247/meetingRooms/283848/keychain* - returns all meeting keys that defined for the meeting room with the meeting number equal to *283848* that assigned to the subscriber with identifier equals to *247*
GET *https://<Wyde bridge domain>/meetings/283848/attributes* - returns all attributes that are set to the meeting room of the meeting with the meeting number equals to *283848*

Resources

Subscriber

This data structure holds information about subscribers. Subscriber is a real person or organization registered on the bridge; the subscriber has a name, phone number, e-mail address, etc. The subscriber can have meeting room info, he does not have access codes, but access codes are properties of meeting keys that belong to the meeting rooms that subscribers have. Subscribers should make a hierarchy – that is why each subscriber has reference to another subscriber who created it. Subscriber which doesn't have a parent - called Administrator. Note that non-admin (non-operator) subscribers can see only "own" information, i.e. his information and information that belongs to subscribers created by him, he can see only their calls, meetings, the reports will show only their data, etc.

Use *https://<Wyde bridge domain>/subscribers* URL to get subscribers data. Subscriber object properties are listed below in Table 1.

`String authenticationToken`	Subscriber's authentication token (encrypted analogue of password), i.e. security token used to prove subscriber's identity; the token is being generated by the bridge on update of the user's password; there are to ways to authenticate the subscriber – either using <login, password> pair, or using <sid, authenticationToken> pair; thus the token is used in addition to or in place of a password to prove that the subscriber is who they claim to be
`DateTime created`	Date and time when record is created; assigned by the server
`String details`	Any additional details, the length is 256 characters max
`String email`	Subscriber's e-mail, the length is 64 characters max
`String eventCallbackUrl`	External URL to be called when an event associated to this subscription is fired; contains URL to sent push-notifications when the subscriber's meetings started, e.g. `` `192.168.1.140/logger/` ``; when the event occurred the server makes http-post to the specified URL and transfers to it the event information in json-format containing: o date/time when the event occurred; o the meeting number that has the event; o the source of the event; o the details about the meeting event (e.g. the meeting was started or ended, the call was started or ended, the recording was started or stopped, etc.) (see details in section "*Event*")
`long externalId`	Subscriber identifier in external system
`String firstName`	Subscriber real first name, the length is 50 characters max
`String lastName`	Subscriber real last name, the length is 50 characters max
`String login`	Login for the logging into the web interface and web services API (*), the length is 50 characters max; login should be unique among all subscribers on the server; if login is used to identify subscriber in a callflow it should consist only of digits
`MeetingRoom[] meetingRooms`	List of all meeting rooms owned by this subscriber, i.e. the meeting rooms this subscriber can possibly attend
`long parentSid`	Sid of the parent subscriber on the bridge (*)
`String password`	Password for the logging into the web interface/API (*), the length is 20 characters max; the password only used during object creation and never returned as a plain text

`String phoneNumber`	Subscriber's phone number used if server needs to dial-out to this subscriber, the length is 24 characters max
`long role`	Subscriber's role (i.e. admin, operator, regular user [default]) (*); possible values: `ROLE_ADMIN (1L)`, `ROLE_OPERATOR (2L)`, `ROLE_USER (3L)`
`long sid`	Unique immutable security identifier (sid) of the subscriber assigned by the bridge

Table 1: Properties of Subscriber

* – for this and all subsequent classes designates mandatory fields during object creation or modification

MeetingRoom (Meeting Room)

This data structure is designed to uniquely identify the meeting and used to manage the meeting attributes. Meeting room is part of subscriber definition (the meeting room info always belongs to the subscriber), thus the subscriber can have multiple meeting rooms, i.e. the meetings the subscriber can possibly attend. Meeting rooms' definition represents subscriber's meeting configuration and contains the meeting number, it defines overridden call flow attributes values exposed by the primary DID group attributes (the attribute's group for meeting's attributes is equal to `MEETING`, see details in section "*CallFlow's Attribute*") as well as contains the meeting keys information representing access code, and role that could be used, and references to DID groups to enter to the given meeting.

Use *https://<Wyde bridge domain>/meetingRooms* URL to get all meeting rooms registered on the bridge. MeetingRoom object properties are listed below in Table 2.

`Attribute[] attributes`	List of attributes and their values imposed by the call flow of the primary DID group this meeting is assigned to; the attribute's group for them is equal to `MEETING`. These attributes are used to override (overwrite) their values for this particular meeting room or the values taken from parent or defaults; they specify MeetingRoom behavior (see details in section "*CallFlow's Attribute*")
`String description`	Description of the meeting; if meetingNumber is omitted holds new meeting description; the length is 256 characters max
`DidGroupReference[] didGroupReferences`	References to DID groups with the phone numbers to be dialed to enter to the given Meeting
`String eventCallbackUrl`	External URL to be called when an event associated to this MeetingRoom is fired; contains URL to sent push-notifications when the meeting of this meeting room is started, e.g. `` `192.168.1.140/logger/` `` (see details about `eventCallbackUrl` field for subscriber in Table 1 and in section "*Event*")
`MeetingKey[] keychain`	List of meeting keys (credentials), i.e. the sets of role and access code to be used to enter to the given MeetingRoom (see details in section "*MeetingKey (Meeting Key for Subscriber's Meeting Room)*")
`long meetingNumber`	The number of the meeting where this user will be assigned after successful authentication. It should be unique across other meeting numbers; 0 means create a new one
`long primaryDidGroupId`	ID of the primary DID Group object that this meeting room is associated with, the primary DID Group defines the set of available call flow attributes overridden for the meeting defined by the meeting room (*)
`String subscriberSid`	The security identifier of the subscriber this meeting room is associated with, i.e. the sid of the subscriber that owns this meeting room

Table 2: Properties of MeetingRoom

MeetingRoom object can exist only if there is the subscriber that owns this meeting room and if this meeting room has the meeting number that is referred by him. The meeting number must be unique across the bridge; different subscribers also can not have the meeting rooms with the same meeting number. The subscriber deletion performs cascade delete of all associated meeting room records.

Additionally, it is possible to override some attributes exposed by default callflow so this meeting room info can define a customized behavior (for example the meeting room info can disable entry tones just for one role while all other users on this number still have them on).

To create a new meeting room definition you need to pass -1 as a meetingNumber or omit this parameter and provide meaningful description of this meeting. In this case server automatically assigns a new unique meetingNumber.

MeetingKey (Meeting Key for Subscriber's Meeting Room)

Meeting key information (MeetingKey) class represents the set of keys used to authorize in the meeting.

It is being used to define a person in a meeting with a particular role (e.g. host, participant, listener, etc.), and the access code that should be entered by the user that called to the meeting to determine his role. A subscriber has meeting rooms associated with him, and each meeting room has meeting keys. Thus a subscriber could be a host user in one meeting and a listener in another.

Meeting key information is the part of the subscriber's meeting room definition and the meeting keys assigned to the same meeting room (with specific meeting number) represent single meeting (meeting room) setup. Please note that meeting key info are not obliged to dial the same DID to get to the same meeting.

MeetingKey object can exist only if there is the meeting room (definition of the meeting) that owns this meeting key. Thus meeting room deletion and subscriber deletion perform cascade delete of all associated meeting key records.

Use *https://<Wyde bridge domain>/subscribers/{ID}/meetingRooms/{meetingNumber}/keychain* URL to get the meeting key data for the specific meeting room (identified by meetingNumber) owned by specific subscriber (identified by his ID). MeetingKey object properties are listed below in Table 3.

`String accessCode`	Access code for the user that should be used after dialing given DID to get into the meeting identified by meeting number. It is used for authentication in a meeting identified by meeting number. Access code should be unique across other accessCodes (*)
`long id`	Unique meeting key identifier assigned by the server
`long role`	Role of this meeting key record: `Host (1L)`, `Participant (2L)`, `Listener (3L)` (*)

Table 3: Properties of MeetingKey

DidGroupReference (DID Group Reference)

DidGroupReference (DID Group Reference) class represents the set of *didGroupId* (DID group identifier) and its *state*. The array of these references is used to determine the DID groups with the phone numbers to be dialed to enter to the given meeting (Meeting Room).

DidGroupReference object properties are listed below in Table 4.

`long didGroupId`	The identifier of DID group that could be used to dial to the meeting (*)
`long state`	This flag determines the status of the DID group: `Active (0L`, default), `Invalidated` / `Inactive (1L)`, `Deleted/Disabled (2L)` (*)

Table 4: Properties of DidGroupReference

CallFlow (Call Flow)

Call flow is a unique meeting service setup registered on the bridge, the logic that is used to process the meeting calls. This is the process a call goes through from call setup to, to processing, to call tear down. It includes the logic, DTMF key-presses used, functions, and the recorded prompts. Each script takes several parameters (like welcome prompt).

Call flows cannot be dynamically created by user as they need to be put into the proper place on the file system and need to be configured by administrator. However end-user should be able to change attributes of already registered call flows in order to customize their behavior.

Use *https://<Wyde bridge domain>/callflows* URL to get call flows data. Call flow object properties are listed below in Table 5.

`Attribute[] attributesTemplates`	Base call flow attribute templates (for DIDs and MeetingRoom info); each attribute template contains the properties described below in Table 6: Properties of Attribute. See details bellow in section "*CallFlow's Attribute*"
`long id`	Unique callflow identifier assigned by the server
`String name`	Callflow description (*), for instance `CONF`, `SPECTEL`, etc.
`String path`	Directory where callflow resides on the server (*)

Table 5: Properties of CallFlow

CallFlow's Attribute

This data structure is used to carry attributes for call flow (CallFlow), DID, DID group (DidGroup) and meeting room (MeetingRoom). The attributes skeleton is defined by call flow. Other entities can only override some of them. So when a user called in to the meeting DID it gets attributes exposed by the call flow. Some of these attributes can be already altered by the DID Group and DID. After the user provided his access code and authentication was successful some attributes can be overwritten again by the attributes of the primary DID Group assigned to the meeting room (MeetingRoom) where the user was authorized and by the attributes of this meeting room.

It is important to remember that list of attributes is always defined by call flow. Values of any attribute may be overwritten by DID Group and MeetingRoom's primary DID Group; depending on the attribute's group (see below) its value could be also overridden by DID or Meeting Room. The hierarchy of the call flow attributes values is either formed as Call Flow → DID Group → DID or it is formed as Call Flow → Meeting Room's Primary DID Group → Meeting Room. Each attribute can be allowed or disallowed for modification by the administrator. The *group* property of the Attribute defines level what the attribute can be overridden at; see Table 6 below. The call flow offers default values for each attribute. For each specific attribute its value usually could be overridden either at DID Group → DID level (*group* property equals *3* in this case) or at Meeting Room's Primary DID Group → Meeting Room level (*group* property equals *5* in this case).

Each attribute has name, type and value. Depending of the type web application should apply one or another validation rule. Also attribute has a "role" so meeting room info can only "see" those attributes which role matches their own role.

Use *https://<Wyde bridge domain>/callflows/{ID}* URL to get the specific call flow data and check its *attributeTemplates* property to get the call flow's attributes data. You can also use the URL *https://<Wyde bridge domain>/meetings/{meetingNumber}/attributes* to get call flow attributes defined for the specific meeting (identified by *meetingNumber*). Attribute object properties are listed below in Table 6.

`String description`	Description of the attribute like "*Exit tones*"
`String enumValues`	If type is *Enum*, i.e. `TYPE_CHOICE (5L)`, this variable holds possible choices like choice1,choice2,choice3 (for example *on,off*), this is read-only field populated by server

`long group`	Meeting access group (role) bitmask flag this attribute belongs to (i.e. the owner of the attribute) consisting of 3 bits for *Call Flow / DID Group*, *DID*, and *Meeting Room* (*); and each of these bits determines at what level the call flow attribute value could be overridden or not: • `bit 0` (*CallFlow* and *DidGroup*) – `0` (0_2) value of this flag determines that the call flow attribute could not be overridden for *call flow* and *DID group*, `1` (1_2) value of this flag determines that the call flow attribute could be overridden for *call flow* and *DID group*; • `bit 1` (*Did*) – `0` (0_2) value of this flag determines that the call flow attribute could not be overridden for *DID*, `1` (1_2) value of this flag determines that the call flow attribute could be overridden for *DID*; • `bit 2` (*MeetingRoom*) – `0` (0_2) value of this flag determines that the call flow attribute could not be overridden for *Meeting Room*, `1` (1_2) value of this flag determines that the call flow attribute could be overridden for *Meeting Room*. For example: o the value *1* (001_2, `ROLE_CALLFLOW`) means that the call flow attribute could be overridden by call flow and DID group; o the value *2* (010_2, `ROLE_DID`) means that the call flow attribute could be overridden by DID; o the value *3* (011_2, `ROLE_CALLFLOW + ROLE_DID`) means that the call flow attribute could be overridden by call flow, DID group and DID; o the value *4* (100_2, `ROLE_MEETING`) means that the call flow attribute could be overridden by meeting room; o the value *5* (101_2, `ROLE_CALLFLOW + ROLE_MEETING`) means that the call flow attribute could be overridden by call flow, DID group and meeting room.
`String name`	Attribute name like "*meeting_exittones*" (*)
`long type`	Attribute type like `TYPE_STRING (0L)`, `TYPE_INT (2L)`, `TYPE_DTMF (3L)`, `TYPE_ROLE (4L)`, `TYPE_CHOICE (5L)` (*)
`String value`	Attribute value like "*on*", "*hp*", "**8*", etc. (*)

Table 6: Properties of Attribute

DID

DID is a unique set of numbers registered on the bridge that is outpulsed by a phone carrier that indicates the intended destination for a particular call. This data structure holds information about registered DID (called phone numbers) on the bridge; it could also contain specific attributes values that override attribute values defined for callflow and DID group. Besides the phone number (usually 10 digits length) each DID has a reference to a DID group.

DIDs of the same call flow if they use the same access codes are being grouped into DID groups; the DID group is a part of the meeting key definition. DID object can exist only if there is the DID group that owns this DID. Thus the DID group deletion performs cascade delete of all grouped DID records.

Use *https://<Wyde bridge domain>/dids* URL to get DIDs data. DID object properties are listed below in Table 7.

`Attribute[] attributes`	DID attributes inherited from callflow; override attribute values defined for callflow and DID group (see details in section "*CallFlow's Attribute*")
`String countryCode`	The region where a phone number is from; ISO 3166-1 two-letter country-code format is used for this field
`String description`	Description of the DID
`long didGroupId`	The identifier of the DID group this DID belongs to
`long id`	Unique DID identifier assigned by the server
`String phoneNumber`	The telephone number or the number pattern ("*", "*712**", "*8665080013*", etc.), or name if connected to VOIP switch (*)
`long state`	This flag determines the status of the DID: `Active (0L)`, `Invalidated` / `Inactive (1L)`, `Deleted/Disabled (2L)`

Table 7: Properties of DID

DidGroup (DID Group)

The DidGroup data structure represents a group of DIDs that belong to the single call flow and used for the specific meeting key record. Different DID groups can be used to connect to the same meeting. DID Group could contain specific attributes values that override attribute values defined for callflow. In addition different DID groups can be based on the same callflows but just have different attributes (like a welcome prompt for example).

Use *https://<Wyde bridge domain>/didsGroups* URL to get the DID groups data. DidGroup object properties are listed below in Table 8.

`Attribute[] attributes`	DID group attributes inherited from callflow; override attribute values defined for callflow; may be overwritten by DID attributes
`long callFlowId`	ID of call flow this DID group belongs to (*); all DIDs from the group belong to the same call flow
`String description`	Description of the DID group
`Did[] dids`	List of DIDs that belong to this group
`long id`	Unique DID group identifier assigned by the server
`String name`	The name of DID group
`long state`	This flag determines the status of the DID group: `Active (0L)`, `Invalidated` / `Inactive (1L)`, `Deleted (2L)`

Table 8: Properties of DidGroup

Meeting

This data structure is used to describe ongoing (active) meeting on the bridge. Objects of this type are only created by server. User may fetch these objects by calling appropriate function. When meeting is over object is deleted by the server.

The meeting object is identified by its number; this is a globally unique identifier, and at any time there could be only one started meeting with the specific meeting number. The meeting number is the property of meeting room info; the meeting keys are assigned to the meeting room with the specific meeting number and all these items determine one single meeting.

Use *https://<Wyde bridge domain>/meetings* URL to get started meetings data. Meeting object properties are listed below in Table 9.

`Attribute[] attributes`	Meeting attributes; the set (template) of available attributes defined for the callflow of the meeting, the attributes values may be overridden on DID and/or MeetingRoom level (see details in section "*CallFlow's Attribute*")
`String broadcastingMode`	Contains URL of the media file (audio or video) that currently is being broadcasted in the meeting, otherwise null or empty string. Note: this field cannot be used as filter criteria, but can be used in sorting.
`DateTime created`	Date and time when this meeting was created – the first caller arrived
`long duration`	Number of seconds which have elapsed since the meeting was created
`String eventCallbackUrl`	Contains URL to sent push-notifications about the meeting events (such as the call started/dropped, the recording started/stopped, etc.), e.g. `` `192.168.1.140/logger/` ``; can be set only when the meeting is started (see details about `eventCallbackUrl` field for subscriber in Table 1 and in section "*Event*")
`long holdMode`	Hold mode bitmask flag consisting of 3 groups for *hosts*, *participants*, and *listeners* and each of the group has bits that determines whether the group is *on hold* or *online* (is not on hold): • *`bits 0,1`* (*host*) – *`0`* (00_2) value of this flag determines that the *hosts* are *online* (not on hold), *`1`* (01_2) value of this flag determines that the *hosts* are *self*-placed *on hold*, *`2`* (10_2) value of this flag determines that the *hosts* are placed *on hold* by the meeting *host*; • *`bits 2,3`* (*participant*) – *`0`* (00_2) value of this flag determines that the *participants* are *online* (not on hold), *`1`* (01_2) value of this flag determines that the *participants* are *self*-placed *on hold*, *`2`* (10_2) value of this flag determines that the *participants* are placed *on hold* by *host*; • *`bits 4,5`* (*listener*) – *`0`* (00_2) value of this flag determines that the *listeners* are *online* (not on hold), *`1`* (01_2) value of this flag determines that the *listeners* are *self*-placed *on hold*, *`2`* (10_2) value of this flag determines that the *listeners* are placed *on hold* by *host*. When the callers are on hold they hear music while hosts have a private discussion. For example: o the value *0* means that the entire meeting is online (nobody is on hold); o the value *40* means that listeners and participants are on hold and were placed on hold by the meeting host.
`boolean isRecording`	This field determines whether the meeting is being recorded
`boolean isSecured`	This field determines whether the meeting is secured, i.e. new calls allowed to join to the meeting or not
`long meetingNumber`	This is an unique meeting number

`long muteMode`	Mute mode bitmask flag consisting of 3 groups for *hosts*, *participants*, and *listeners* and each of the group has bits that determines the specific mute mode *open*, *relaxed*, and *strict*: • `bits 0,1` (*host*) – `0` (00_2) value of this flag determines that the *host* mute mode is *open* (is not muted), `1` (01_2) value of this flag determines that the *host* mute mode is *relaxed*, `2` (10_2) value of this flag determines that the *host* mute mode is *strict*; • `bits 2,3` (*participant*) – `0` (00_2) value of this flag determines that the *participant* mute mode is *open* (is not muted), `1` (01_2) value of this flag determines that the *participant* mute mode is *relaxed*, `2` (10_2) value of this flag determines that the *participant* mute mode is *strict*; • `bits 4,5` (*listener*) – `0` (00_2) value of this flag determines that the *listener* mute mode is *open* (is not muted), `1` (01_2) value of this flag determines that the *listener* mute mode is *relaxed*, `2` (10_2) value of this flag determines that the *listener* mute mode is *strict*. When the callers mute mode is *relaxed* they are muted, but the audience can un-mute themselves. When the callers mute mode is *strict* they are muted, but the audience does not have the capability of un-muting themselves, i.e. they cannot un-mute themselves. For example: o the value *32* means that the listeners are muted and cannot un-mute themselves, participants and hosts are not muted; o the value *36* means that the listeners are muted and cannot un-mute themselves, participants are muted and can un-mute themselves, and hosts are not muted (mute mode – *relaxed*); o the value *40* means that the listeners and participants are muted and cannot un-mute themselves, hosts are not muted (mute mode – *strict*).
`OperatorStatus operatorStatus`	This fields represents the operator's activity, i.e. it contains the data structure that describes the operator's meeting (see details in section "*OperatorStatus (Operator Status)*")
`long participantCount`	Number of active participants in the meeting
`String pollingMode`	This field determines whether the polling call is started; *empty* string value means that the polling is *not started*; if the polling is *started* the field contains available polling options – digits 1, 2, ..., 9, 0
`long qaMode`	This field determines whether the Q&A call is started and used to manage Q&A mode; *0* value of this flag means that the Q&A call is *not started*, *1* value of this flag means that the Q&A call is *started* (in progress), *2* value means *talk to the next* caller in Q&A queue, *4* value means *clear Q&A queue*
`long state`	The status of the meeting: *0* – the meeting completed or is not started, *1* – the meeting is on hold (for instance, due the moderator is not arrived yet when it is required by the meeting configuration, see *How conference begins* (`conference_start_how`) call flow attribute value), *2* – the meeting is active Note: there is no transition from the state *2* (active) to the state *1* (on hold)
`String subscriberName`	Name of the first subscriber in the meeting
`long type`	This field determines whether the meeting is participating in the distributed meetings (DC, the value *1*) or not (the value *0*)

Table 9: Properties of Meeting

OperatorStatus (Operator Status)

This data structure is designed to show the status of the operator's meeting.

Use *https://<Wyde bridge domain>/meetings/{meetingNumber}* URL to get the specific meeting data and check its *operatorStatus* property to get the status of the operator's meeting for the specified meeting. OperatorStatus object properties are listed below in Table 10.

`Long engagedConferenceNumber`	Meeting number of the connected meeting
`boolean isConnected`	This field determines whether the operator's meeting is currently connected to the other one (in this case this property is set to true).
`boolean isMonitoring`	For the operator meeting this field determines whether the operator meeting is in scanning mode (i.e. surveillance call, usually started when the operator presses *1 on his phone keypad)
`long status`	This field determines operators meeting mode `MEETING_REGULAR (0L)`, `MEETING_OPERATOR (1L)`, `MEETING_LISTEN (2L)`, `MEETING_AUTOLISTEN (3L)`, `MEETING_AUTOLISTEN_SLEEP (4L)`, `MEETING_TALK (5L)`

Table 10: Properties of OperatorStatus

Call

This data structure represents a single ongoing (active) call on the server. User can not directly create this object. It is being created by the server when the user calls on the bridge. When the call is over server automatically deletes this object.

Normally this data structure is used to get information about call attributes like calling/called number etc. If something needs to be done with the call (mute/hang/hold) the call should be referenced by its identifier.

Use *https://<Wyde bridge domain>/calls* URL to get started calls data. Call object properties are listed below in Table 11.

`String accessCode`	Access code entered by caller
`String addressFrom`	Full address FROM, i.e. full qualified caller's address
`String addressTo`	Full address TO, i.e. full qualified callee's address
`String bridgeName`	Name of hosted bridge
`String callee`	Information about callee as it is provided in TO field
`String caller`	Information about caller as it is provided in FROM field (normally the phone number)
`String codec`	The active audio codec of the call, i.e. the technical name of the codec, that is used by the call, for example PCMU for uLaw, SIREN7 for 722.1, SIREN14 for 722.2, ILBC, etc.
`long connectionStatus`	Bit-mask that indicates the optional call attributes, for example call direction: 0 – the inbound call, 1 – the outbound call
`DateTime created`	Date and time when this call was created
`String customName`	Custom user name either set from the web or IVR (login)
`long duration`	Number of seconds which have elapsed since the call started

`long gainLevel`	The microphone volume level of the call, i.e. gain control option; it could be from -10 till 10 or 255; -10 is the quietest (lowest) sound level, 10 is the loudest (highest) sound level, 255 denotes that the microphone level is being automatically adjusted by the backend
`long holdMode`	Hold mode flag that determines whether the call is put on hold (by administrator, owner, client, etc.) or the call is *online* (is not on hold): *0* value of this flag determines that the call is online, *1* value of this flag determines that the call is on hold
`long id`	Unique ID assigned by the call
`String jobCode`	Active billing (business) code
`DateTime joined`	Time when this call joined to the meeting
`long meetingNumber`	Meeting number of the meeting this call belongs to
`long muteMode`	Mute mode bitmask flag consisting of 2 groups (bits) for *host muting* and *self muting*; and each of the group has bit that determines whether this call is muted or not: • `bit 0` (*host*) – `0` (0_2) value of this flag determines that the call is not muted by the *host*, `1` (1_2) value of this flag determines that the *host* muted the call; • `bit 1` (*self*) – `0` (0_2) value of this flag determines that the caller *self* is not muted the call, `1` (1_2) value of this flag determines that the caller *self* muted the call
`String nodeName`	Name of hosted node
`long operatorMode`	This filed represents the operator's activity (for instance, empty, waiting for operator, speaking with operator, etc.). Possible values: • `0L` (or `null`) – the caller does not need operator assistance; • `1L` (`wait`) – the caller is waiting operator assistance, i.e. the caller is in the operator's queue; • `2L` (`talk`) – the caller is talking to the operator
`long presenterMedia`	Specific media of the call chunk; possible values: *audio* (1L), *screensharing* (2L), *video* (4L), *controlling* (8L)
`long qaMode`	This filed represents Q&A mode for current call: `QA_STATUS_IDLE (0L)`, `QA_STATUS_RAISEDHAND (1L)`, `QA_STAUS_ACTIVE (2L)`
`long role`	This field determines what role this call has. The roles should be the same as in MeetingKey. Role helps to verify whether this call is allowed to do recording – `MODE_UNDEFINED (0L)` `MODE_HOST (1L)` – host permissions granted, `MODE_PARTICIPANT (2L)` – caller controls muting, i.e. the call owner can mute/unmute himself, `MODE_LISTENER (3L)` – the call owner can only listen and can not talk, `MODE_RECORDING (4L)` – the recording call, `MODE_DC_LINK (8L)` – distributed meeting (DC) link, i.e. the control call between two bridges in distributed conferencing
`long state`	This field determines whether the current call state: `STATUS_IVR (1L)` - call is owned by frontend; `STATUS_MEETING (2L)` - call is owned by backend; `STATUS_CLOSED (3L)` - call is closed; `STATUS_DIALING (4L)` - call is dealing
`String subconference`	If non-empty, denotes current sub-meeting of the call (caller)
`String subscriberPin`	Contains subscriber PIN if it was entered by the user from his DTMF keypad when he joins to the meeting to receive host permissions in the meeting; check call flow attributes *Validate subscriber pin* (`dnis_validatesubscriberpin`) and *Subscriber PIN* (`subscriber_pin`) for details

Table 11: Properties of Call

Mdr (Meetings Detail Records)

This data structure is used to describe meeting which is already terminated on the bridge. User can not directly create this object.

Use *https://<Wyde bridge domain>/mdrs* URL to get started terminated meetings data. Mdr object properties are listed below in Table 12.

`long audioParticipantCount`	Number of audio participants (i.e. made voice calls) in the meeting
`long controlParticipantCount`	Number of control calls in the meeting
`DateTime created`	Time when this meeting was created - first caller arrived
`CustomExtension[] customExtensions`	List of available (generated) custom extensions of screen sharing recordings (see "*CustomExtension (Custom Extension)*" section for details)
`String description`	Meeting description
`long duration`	Number of seconds which have elapsed since the meeting was created till the time when it was terminated
`DateTime expirePeriod`	Expiration period for shared recording URL
`long fileSize`	The size of generated persistent meeting recording files, e.g. g722/ul, mp3-file size plus flv-file size (if screen sharing was recorded); this does not include generated custom files size; if your screen sharing recording was converted to the custom extension (for example mp4) to get the total size of your meeting recordings you should also add fileSize for all generated custom extension files (check *customExtensions* property)
`boolean hasPollingResults`	Whether or not the meeting was voted
`boolean hasRecording`	Whether or not the meeting was recorded
`String jobCode`	Active billing (business) code of the meeting
`long meetingId`	Unique meeting identifier assigned by the server
`long meetingNumber`	This is a meeting number
`long participantCount`	Number of participants in the meeting
`String password`	Meeting recordings password (6 digits maximum), if the recording password is set it should be entered to download and/or playback the recording
`long processingStatus`	Screen sharing video flv-file conversion status (from 0 till 100, or -1 in case of any processing error), i.e. creating the flv-file from the raw format
`long recordingDuration`	Meeting audio recording duration in seconds
`String recordingUrl`	URL for the audio recording
`long referenceNumber`	Serial (consecutive) unique meeting recording reference number (counter from 1), the number related to the specific recording is consistent and never changed
`long screenSharingParticipantCount`	Number of screen sharing participants (participated in screen sharing) in the meeting
`String sharedRecordingUrl`	URL for the shared audio recording
`long videoParticipantCount`	Number of video participants (i.e. made video calls) in the meeting
`long webRecordingDuration`	Meeting video duration in seconds

Table 12: Properties of Mdr

CustomExtension (Custom Extension)

This data structure represents custom extensions of screen sharing recordings. Default screen sharing video format is *flv*. Currently custom extension *mp4*-format only is supported.

Use *https://<Wyde bridge domain>/mdrs* URL to get information about terminated meetings data and check its *customExtensions* property to get the list of available (generated) custom extensions of screen sharing recordings for the specified meeting. CustomExtension object properties are listed below in Table 13.

`String extension`	Extension of generated custom video meeting recording file (for example mp4)
`long fileSize`	Size of generated meeting recording file in the specific custom format (extension)
`long processingStatus`	Screen sharing custom video file (for example *mp4*) conversion status (from *0* till *100*, or *-1* in case of any processing error)

Table 13: Properties of CustomExtension

Cdr (Calls Details Records)

This data structure represents a single call on the server which is already terminated on the on the bridge. User can not directly create this object.

Note if the operator was involved into the call – the user called to the operator and the operator attached the user to another meeting there would be two Cdr records with the same call detail record identifier (callId). These records will differ by disconnect reason.

Use *https://<Wyde bridge domain>/cdrs* URL to get started terminated calls data. Cdr object properties are listed below in Table 14.

`String accessCode`	Access code entered by caller
`String addressFrom`	Full address FROM, i.e. full qualified caller's address
`String addressTo`	Full address TO, i.e. full qualified callee's address
`long audioDuration`	Duration in seconds of how long the audio (voice) call was active, i.e. the total duration of all *audio* media chunks for the call
`String bridgeName`	Name of hosted bridge
`String callee`	Information about callee as it is provided in TO field
`String caller`	Information about caller as it is provided in FROM field (normally the phone number)
`long callId`	Unique call ID assigned by the server
`long controllingDuration`	Duration in seconds of how long the control call was active, i.e. the total duration of all *controlling* media chunks for the call
`DateTime created`	Time when this call was created
`String customName`	Custom user name either set from the web or IVR (login)
`String customType`	Custom call type: • `Foreign` – the 3rd party VoIP clients; • `Private` – the native WYDE clients, i.e. the clients that are using WYDE VoIP library; • `Private-ASD` – the native WYDE real-time client; • `Recording` – the meeting recording service call
`long direction`	Call direction: 0 – the inbound call, 1 – the outbound call
`long disconnectInitiator`	Shows who initiated a disconnect (user, bridge): `INITIATOR_BRIDGE (2L)` – used when the call was terminated by bridge; `INITIATOR_UNDEFINED (0L)` – used when initiator is not defined; `INITIATOR_USER (1L)` – used when the call was terminated by user

`String disconnectReason`	A string showing detailed info about disconnect
`long duration`	Number of seconds which have elapsed since the call started and before disconnect
`CallEvent[] events`	List of DTMF call events occurred in the specific call (see "*CallEvent (DTMF Call* Event" section for details)
`String jobCode`	Active billing (business) code
`DateTime joined`	Time when this call joined to the meeting
`long meetingId`	Meeting identifier of the meeting this call belongs to
`long meetingNumber`	Meeting number this call belongs to
`String nodeName`	Name of hosted node
`long role`	This field determines what role this call had.
`long screenSharingDuration`	Duration in seconds of how long the screen sharing session was active, i.e. the total duration of all *screensharing* media chunks for the call
`String subscriberId`	The security identifier of subscriber assigned by the call if it could be determined (for example if the call is connected to free call flow without authorization, this identifier would be undefined)
`long videoDuration`	Duration in seconds of how long the video call was active, i.e. the total duration of all *video* media chunks for the call

Table 14: Properties of Cdr

CallEvent (DTMF Call Event)

This data structure represents a single DTMF command occurred in the specific call.

Use *https://<Wyde bridge domain>/callEvents* URL to get DTMF call events data. CallEvent object properties are listed below in Table 15.

`String action`	"`DTMF`" string constant identifying the data represent DTMF command.
`long callId`	The identifier of the call where DTMF event occurred
`DateTime created`	Date and time when this DTMF event was initiated (occurred)
`String details`	Specific DTMF command, for example "*#||110953#*"
`long id`	Unique DTMF call event identifier assigned by the server

Table 15: Properties of CallEvent

Event

This data structure represents the session event logger, i.e. it represents the events occurred and logged on the bridge (for example: meeting created, meeting dropped, as well as real time event notifications when call created, call dropped, recording started, recording stopped, etc.).

Note if *eventCallbackUrl* property is set either for Subscriber or Meeting Room or Meeting, when the event occurred the server makes http-post to the URL specified in this property and transfers (sends push-notifications) to it the event information in json-format containing the occurred event data described in Table 16.

Use *https://<Wyde bridge domain>/events* URL to get event logger data. Event object properties are listed below in Table 16.

`DateTime created`	Date and time when this event was created (occurred)
`String details`	Detail information about occurred event, e.g. `CONF_CREATE` and `CONF_DROP` for *`Subscription`* source or real time notifications (for example "`NOTIFY-GROUP Listener MUTE Strict HOLD False`") for *`Meeting`* source; see "*Real Time Interface – Programmer's Guide*", section "*Notification Format*" for detail information about possible RT event notifications
`long meetingNumber`	Meeting number this event is applied to, i.e. the meeting that has this event
`String source`	The source of the event: • *`Subscription`* – if the event came from the subscription; • *`Meeting`* – if the event came from the Real Time Interface notifications

Table 16: Properties of Event

Settings' Attribute

This data structure is designed to retrieve and manage bridge settings. You can list all or specific WYDE bridge settings as well as define the value for the setting's attribute.

Use *https://<Wyde bridge domain>/settings* URL to get bridge settings. Settings' attribute object properties are listed below in Table 17.

`String description`	Always empty (omitted) for the settings' attributes
`String enumValues`	Always empty (omitted) for the settings' attributes
`long group`	Always empty (omitted) for the settings' attributes
`String name`	Settings attribute (parameter) name like "*agiserver_addr*", "*billing_localdb_driver*", "*mf_dc*", etc. (*)
`long type`	Always string attribute type as `TYPE_STRING (0L)`
`String value`	Settings attribute (parameter) value like "*10.1.3.140:5050*", "*pgsql*", "*0*", etc. (*)

Table 17: Properties of Settings' Attribute

Bridge

This data structure is designed to retrieve information about the current (local) bridge. Note: you can retrieve information about local bridge without authorization.

Use *https://<Wyde bridge domain>/bridge* URL to get your local bridge info. Bridge object properties are listed below in Table 18.

`long dcPort`	Distributed conferencing port number, e.g. *4470*
`long id`	Unique identifier of the bridge assigned by the server
`String ipAddress`	IP address of the bridge, e.g. "*192.168.1.32*"
`String name`	Name of the bridge
`long role`	Role of the bridge: `0L` – local bridge, `2L` – distributed conferencing bridge
`Attribute[] settings`	List of settings for the bridge (see "*Bridge*" section for details)
`long sipPort`	SIP port of the bridge, e.g. *5060*
`String version`	Version of the bridge, e.g. "*4.0.84*"

Table 18: Properties of Bridge

Bridges

This resource is designed to retrieve information about all registered bridges.

The data structure represents the list (array) of Bridge objects described the section above and in Table 18. Use *https://<Wyde bridge domain>/bridges* URL to get information about all registered bridges and their settings.

HTTP Methods

WYDE REST web services use four HTTP methods: GET, POST, PUT, and DELETE. See Table 19 below for detailed information.

HTTP method	Collection (e.g. /subscribers) returns	Single object (e.g. /subscribers/247) returns
GET - used to retrieve a representation of a resource	*200 OK* – returns list of objects according to supplied range, filter and order *400 Bad Request* – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format *500 Internal server error* (see below)	*200 OK* – returns single object according to supplied identifier *404 Not Found* – object with specified identifier is not found *500 Internal server error* (see below)
POST – used to create a new object	*201 Created* – the object was created; returns the URL (Location header) to the created object (e.g. *https://<Wyde bridge domain>/subscribers/278*) *400 Bad Request* – the object was not created; for example if wrong data were specified (e.g. wrong field name or field value were specified, mandatory parameter missing, etc.); short error description would be supplied in the plain text format *500 Internal server error* (see below)	–
PUT – used to update existing object	–	*200 OK* – the object was updated; returns the URL (Location header) to the updated object (e.g. *https://<Wyde bridge domain>/subscribers/247*) *400 Bad Request* – the object was not updated (for example the object with such identifier is not found); short error description would be supplied in the plain text format *404 Not Found* – object with specified identifier is not found *500 Internal server error* (see below)
DELETE - used to delete a resource	–	*204 No Content* – the object was successfully deleted; *404 Not Found* – the object with specified identifier is not found *500 Internal server error* (see below)

Table 19: Using of HTTP Methods

Note: for any HTTP method *500 Internal server error* represents a generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

The examples of the HTTP methods calls are below:
GET *https://<Wyde bridge domain>/subscribers*
GET *https://<Wyde bridge domain>/subscribers/247*
GET *https://<Wyde bridge domain>/subscribers/247/meetingRooms*
GET *https://<Wyde bridge domain>/subscribers/247/meetingRooms/283848*
GET *https://<Wyde bridge domain>/subscribers/247/meetingRooms/283848/keychain*

POST *https://<Wyde bridge domain>/subscribers/*
POST *https://<Wyde bridge domain>/subscribers/247/meetingRooms*

PUT *https://<Wyde bridge domain>/subscribers/247*
PUT *https://<Wyde bridge domain>/subscribers/247/meetingRooms/283848*

DELETE *https://<Wyde bridge domain>/subscribers/247*
DELETE *https://<Wyde bridge domain>/subscribers/247/meetingRooms/283848*

Errors

If the object creation method POST or the object modification method PUT returns the error "*400 Bad Request*" the ErrorResponse object would be returned. Error response properties are listed below in Table 20.

`long status`	The status of the error, e.g. *400* for the "*Bad Request*" error
`String message`	The general description of the error, e.g. "*Object validation error*"
`ErrorDetails[] errors`	List of error details information for all errors occurred: • `Field` – the name of the object field that contains invalid value (optional, if the error is related to the specific field); • `Message` – the error description for the specific field (mandatory)

Table 20: Properties of ErrorResponse

```
{
  "errors": [
    {
      "field": "callFlowId",
      "message": "Wrong value"
    },
    {
      "field": "meetingNumber",
      "message": "Wrong value"
    }
  ],
  "message": "Object validation error",
  "status": 400
}
```

Figure 1: Error Sample

Returning Representations

Web services currently support JSON representations of resources only (no XML support as of now). Web services support the HTTP Accept header and the file-extension-style format identifier (specification). Default web services results format is JSON, but you must explicitly specify *Response Content Type* equal to *application/json*.

To return data in JSON format you should use *.json* suffix in the web services call URL or omit the format identifier suffix. For example:
GET *https://<Wyde bridge domain>/meetings*
GET *https://<Wyde bridge domain>/meetings.json*
GET *https://<Wyde bridge domain>/meetings/49843764*
GET *https://<Wyde bridge domain>/meetings/49843764.json*

For example the following command:
`curl -v "http://192.168.1.32/wyderef/meetings/49843764.json"`
returns specific meeting (with meeting number 49843764) data in JSON format as shown on Figure 2.


```
{"attributes":[{"group":0,"name":"conference_entrytones","value":"
off"}],"broadcastingMode":"","created":"2014-11-11
12:40:15","duration":9932,"holdMode":0,"id":125,"isRecording":fals
e,"isSecured":false,"meetingNumber":49843764,"muteMode":32,"partic
ipantCount":1,"qaMode":-1,"state":2,"type":1}
```


Figure 2: Specific Started Meeting Data in JSON Format

Response Headers

The sample of WYDE REST web services call is shown in Figure 3. The response message consists of elements listed in Table 21.

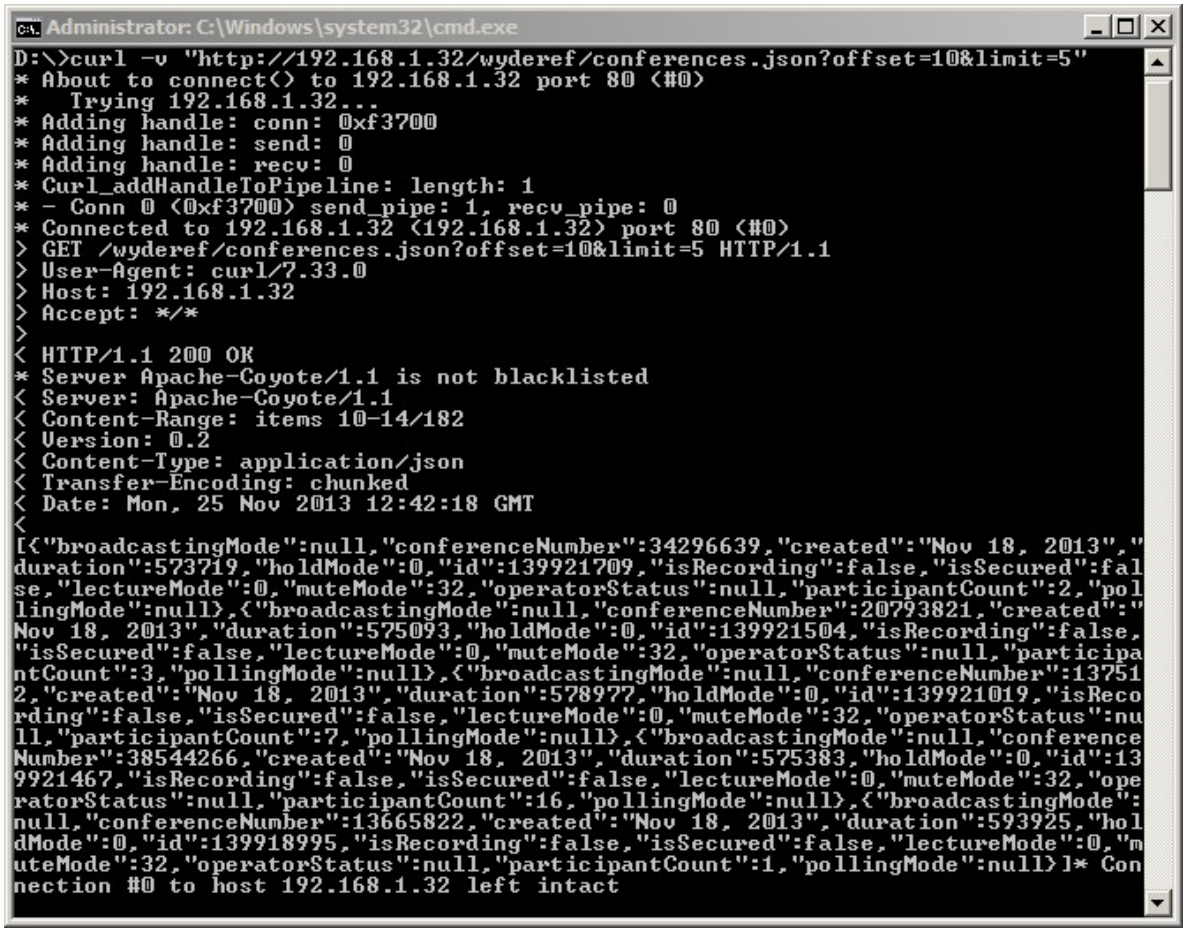

Figure 3: Web Services Response Sample

Identifier	Samples	Description
Status-Line	HTTP/1.1 200 OK	HTTP response status codes[3] as described in Table 19; this indicates that the client's request is succeeded or failed
Server	Server: Apache-Coyote/1.1 Server: nginx/0.8.55	The version of the server
Content-Range	Content-Range: items 10-14/182	The range of records that were returned and total number of records in form: *<zero-based first returned record>-<zero-based last returned record>,<total number of records>*
Content-Type	Content-Type: application/json	The Internet media type – a standard identifier used on the Internet to indicate the type of data that a file contains; for instance: *application/json*, *application/xml* (currently not supported)

[3] See details at http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Version	Version: 1.0.5	The version of WYDE REST web services. Note: it is not the version of the WYDE bridge software installed
Transfer-Encoding	Transfer-Encoding: chunked	Chunked transfer encoding is a data transfer mechanism in version 1.1 of the HTTP in which data is sent in a series of "chunks"[4]
Connection	Connection: keep-alive	What type of connection the user-agent would prefer; for instance: *keep-alive*
Access-Control-Allow-Origin	Access-Control-Allow-Origin: *	Specifies which web sites can participate in cross-origin resource sharing; for instance: * (used to allow access from all domains)
Access-Control-Allow-Methods	Access-Control-Allow-Methods: GET, POST, DELETE, PUT	Specifies the methods allowed when accessing the resource; for instance WYDE web services support the following methods: GET, POST, DELETE, PUT
Access-Control-Allow-Headers	Access-Control-Allow-Headers: Content-Type	Used to indicate which HTTP headers can be used when making the actual request; for example *Content-Type* header is used to denote that *application/json* should be used for output data
Date	Date: Thu, 02 Oct 2014 10:27:23 GMT	Date and time when the request was made

Table 21: Elements of Web Services Response Header

Pagination

WYDE REST web services use the URL parameters *offset* and *limit* to specify what data should be returned:
`offset` – long zero-based offset in recordset that specifies the first record that should be returned;
`limit` – long maximum number of objects to return.

For example use the call:
GET *https://<Wyde bridge domain>/meetings.json?offset=10&limit=5*
to return 5 meetings (parameter *limit=5*) starting from the 11th meeting (parameter: *offset=10*, note the offset is zero-based).

As it as was previously mentioned, the response header will contain the *Content-Range* token, specifying the actual range of the returned records and total number of records.
For example in our sample it would be:
Content-Range: items 10-14/182
that means that the meetings from *10*th till *14*th are returned and total number of the started meetings on the bridge is *182*.

[4] See details at http://en.wikipedia.org/wiki/Chunked_transfer_encoding

Filtering and Sorting

In WYDE REST web services filtering and sorting are made using *filter* and *order* parameter that should be specified in the web services call URLs.

Filtering

A URL with a filter identifies a subset of the entries that should be returned by the web service call. The subset is determined by selecting only the entries that satisfy the predicate expression specified by the query filter option. WYDE REST web services use the single *filter* parameter, for example:
GET *https://<Wyde bridge domain>/subscribers.json?filter=login%20eq%20admin*
GET *https://<Wyde bridge domain>/subscribers.json?filter="login eq admin"*

The expression language that is used in filter operators supports references to properties and literals. The literal values can be strings enclosed in single quotes, numbers and Boolean values (true or false).

The operators supported in the expression language are shown in the Table 22 below; the operators are case insensitive. Note: or-clauses and brackets are not supported.

Operator	Description	Old	Example
eq	Equal	=	`/subscribers?filter="login eq 'admin'"`
ne	Not equal	!=	`/subscribers?filter="role ne 3"`
gt	Greater than	>	`/subscribers?filter="role gt 1"`
ge	Greater than or equal	>=	`/subscribers?filter="role ge 2"`
lt	Less than	<	`/subscribers?filter="id lt 100"`
le	Less than or equal	<=	`/subscribers?filter="parentId le 10"`
and	Logical and	and	`/subscribers?filter="parentId eq 1 and role gt 1"`
like	Case-sensitive like	like	`/subscribers?filter="firstName like 'A%'"`
ilike	Case-insensitive like	ilike	`/subscribers?filter="firstName ilike '%admin%'"`

Table 22: List of Supported Filter Operators

Sorting

WYDE REST web services use the single *order* parameter to define the order of returned set of records, for example:
GET *https://<Wyde bridge domain>/subscribers?order=role%20desc*
GET *https://<Wyde bridge domain>/subscribers?order="role desc"*

The default direction is ascending and *asc* can be omitted, for descending order use *desc*. Currently you can sort by the single field value only; no multiply fields sorting supported as of now.

Real Time Interface over Web Sockets

WYDE Conferences Bridge also provides the possibility to work with the real time interface over Web Sockets[5]. You can use the URL: *wss://<Wyde bridge domain>/wyderef/websocket/meetings/{meetingNumber}* to establish a Web Socket connection to the specific meeting *{meetingNumber}* (for example *wss://192.168.1.32/wyderef/websocket/meetings/614695*). This connection provides full-duplex communication channel over a single TCP connection, i.e. it allows communication (streams of messages) in both directions – the bridge supports receiving RT requests and sending RT responses / notifications. Once you opened the Web Socket connection to the specific meeting you will receive the meeting's RT notifications as well as will be able to send RT commands to this meeting.

When you would like to execute any command you should send the request message to the bridge. Once the request is received by the bridge, the response message is being generated and returned; the response message contains the identifier of your request and the response code (error code). Requested commands are being executed asynchronously; when the requested command or any other conference event is occurred the notification message will be sent by the bridge back to you. See "*Real Time Interface – Programmer's Guide*" for additional information about WYDE bridge RT interface, request commands, and RT notifications.

See Figure 4 for the sample how to work with RT interface over Web Sockets.

```
<html>
  <head>
    <script>
      // open Web Socket connection to the specific meeting
      var webSocket = new
            WebSocket('wss://192.168.1.32/wyderef/websocket/meetings/614695');
      // implement some code on opening Web Socket connection
      webSocket.onopen = function(event) {
        // some business logic goes here, for example mute the meeting:
        webSocket.send("MUTE-GROUP 1435588081108 Relaxed Speaker");
      };
      // process receiving notifications from the meeting
      webSocket.onmessage = function(event) {
        alert('onmessage, ' + event.data);
      };
      // process closing of Web Socket connection
      webSocket.onclose = function(event) {
        // some business logic may go here...
      };
    </script>
  </head>
  <body>
  </body>
</html>
```

Figure 4: Sample of JavaScript Code to Work with Real Time Interface over Web Sockets

[5] See details about WebSocket protocol at https://en.wikipedia.org/wiki/WebSocket

See Figure 5 and Figure 6 below for the samples of headers of an opened Web Socket connection to the specific meeting and Real Time responses and notifications from this meeting received over Web Socket connection.

Figure 5: Sample of Headers Information about Web Socket Connection to a Meeting

Headers | Frames | Cookies | Timing

Data	Leng...	Time
NOTIFY-VERSION 4.0.86	21	17:30:12.799
NOTIFY-CONFERENCE 614695 134217732 False False False 1435587163	63	17:30:12.801
NOTIFY-JOIN 16777223 audio Foreign 0 Moderator False False False False False False True 1435587158 12 "" "'unknown'" 0 0	126	17:30:12.801
NOTIFY-JOIN 16777224 audio Foreign 0 Speaker False False False False False False False False 1435587166 12 "" "'unknown'" 0 0	125	17:30:12.801
NOTIFY-GROUP Moderator MUTE False HOLD False	44	17:30:12.801
NOTIFY-GROUP Speaker MUTE False HOLD False	42	17:30:12.801
NOTIFY-GROUP Listener MUTE Strict HOLD False	44	17:30:12.802
NOTIFY-ENDOFJOINING	19	17:30:12.802
NOTIFY-VERSION 4.0.86	21	17:31:24.782
NOTIFY-CONFERENCE 614695 134217732 False False False 1435587163	63	17:31:24.862
NOTIFY-JOIN 16777223 audio Foreign 0 Moderator False False False False False False True 1435587158 12 "" "'unknown'" 0 0	126	17:31:24.862
NOTIFY-JOIN 16777224 audio Foreign 0 Speaker False False False False False False False False 1435587166 12 "" "'unknown'" 0 0	125	17:31:25.379
NOTIFY-GROUP Moderator MUTE False HOLD False	44	17:31:25.380
NOTIFY-GROUP Speaker MUTE False HOLD False	42	17:31:25.380
NOTIFY-GROUP Listener MUTE Strict HOLD False	44	17:31:25.380
NOTIFY-ENDOFJOINING	19	17:31:25.380
MUTE-GROUP 1435588081108 Relaxed Speaker	40	17:40:34.768
QA-MODE 1435588081109 False	27	17:40:34.769
RESPONSE 1435588081108 0	24	17:40:34.898
NOTIFY-GROUP Speaker MUTE Relaxed HOLD False 29503584403457	59	17:40:35.065
RESPONSE 1435588081109 0	24	17:40:35.115
NOTIFY-MUTE False Relaxed False True 16777224	45	17:40:35.652
MUTE-GROUP 1435588081110 False Speaker	38	17:41:50.573
QA-MODE 1435588081111 False	27	17:41:50.575
RESPONSE 1435588081110 0	24	17:41:50.888
NOTIFY-GROUP Speaker MUTE False HOLD False 29503584403458	57	17:41:50.895
NOTIFY-MUTE False False False False 16777224	44	17:41:50.927
RESPONSE 1435588081111 0	24	17:41:50.936

Figure 6: Sample of RT Responses and Notifications Received over Web Socket Connection

Chapter 3: Function Reference

The samples of the functions below are provided using *cURL*[6] command-line tool. The following format is used for testing:

curl -v -u <subscriber's login>:<subscriber's password> [-X POST –data <values>]
"*https://<Wyde bridge domain>/*
<method / resources>[/<identifier>][?<parameter1>[<parameter2>[...]]]"

Note 1: to escape special characters in parameters use URL encoding (percent-encoding) approach[7]; for example you should replace the *space* with `%20` and the *double quotes* (") with `%22` and the *percent sign* (%) with `%25`.

Note 2: to filter by data containing spaces enclose these data into the *single quotes* – apostrophes (').

Subscribers Management

- `GET https://<Wyde bridge domain>/subscribers` – This method returns the list of subscribers which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output. Please note that field *meetingRooms* in Subscriber will be populated with the list of meeting rooms this subscriber associated with and can possibly attend.
  Parameters:
  - `offset` – long zero based offset in recordset.
  - `limit` – long maximum number of objects to return.
  - `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more subscriber field names (any subscriber fields are permitted); additionally to filter subscriber data you can use subscriber's meeting rooms fields (e.g. meetingRooms.meetingNumber) as well as meeting keys fields of subscriber's meeting rooms (e.g. meetingRoom.meetingKey.accessCode). Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.
  - `order` – string specifying any single subscriber field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.
  - `includeFields` – comma separated list fields that should be included in the response (e.g. *login,role*).
  - `excludeFields` – comma separated list fields that should be excluded from the response (e.g. *meetingRooms.keychain,meetingRooms.attributes* or *meetingRooms*).

[6] See details at http://en.wikipedia.org/wiki/CURL
[7] See details at http://en.wikipedia.org/wiki/Percent-encoding

Returns Data:

list of Subscriber objects

Returns Status:

200 OK – returns list of Subscriber objects with the specified fields presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers"	Return all subscribers with all fields
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers?offset=3&limit=3&order=login%20desc&includeFields=login%2Crole%2CphoneNumber"	Return *3* subscribers *4th* till *6th*; the objects should be sorted by their *login* descending; only *login*, *role*, and *phoneNumber* fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers?filter=role%20eq%203&order=login&excludeFields=meetingRooms"	Return subscribers with user (`3L`) *role*; the objects should be sorted by their *login* ascending; no *meetingRooms* fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers?filter=role%20le%202&order=login&excludeFields=meetingRooms.keychain%2CmeetingRooms.attributes"	Return subscribers with admin (`1L`) and operator (`2L`) *role*; the objects should be sorted by their *login* ascending; *meetingRooms.keychain*, *meetingRooms.attributes* fields should be excluded from the response
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers?filter=login%20eq%20admin"	Return the subscriber with *login* equals to 'admin'; include all fields
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers?filter=firstName%20like%20'A%25'&order=firstName"	Return subscribers with the *first name* started with 'A' ordered by *first name* ascending; include all fields
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers?filter=meetingRoom.meetingNumber%20eq%20111"	Return subscribers that have meeting rooms with *meeting number* equal 111; include all fields
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers?filter=meetingRoom.meetingKey.accessCode%20eq%20111"	Return subscribers that have meeting rooms with meeting keys with *access code* equal 111; include all fields

Note:

This method replaces `getSubscribers (long offset, long limit, String filter, String order)` from SOAP API.

- `GET https://<Wyde bridge domain>/subscribers/{SID}` – This method returns the full details about the subscriber referenced by the security identifier *{SID}* provided.

Parameters:

`SID` – The subscriber security identifier

`includeFields` – comma separated list fields that should be included in the response (e.g. *login,role*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *meetingRooms.keychain,meetingRooms.attributes* or *meetingRooms*).

Returns Data:

Subscriber object

Returns Status:

200 OK – returns single Subscriber object according to supplied identifier

404 Not Found – object matched to the criteria is not found, i.e. the subscriber with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7"	Return the details for the subscriber with *security identifier* '*S91s93FslPe-y0.6sAW2XMKp8IYqxml7*'

Note:

This method replaces ***`getSubscriber`*** `(long subscriberId)` from SOAP API.

- ***`GET`*** ***`https://<Wyde bridge domain>/subscribers/whoami`*** – This method returns the details about the current active subscriber, i.e. the subscriber used in the web services authorization.

  Returns Data:

  Subscriber object

  Returns Status:

  200 OK – returns single Subscriber object according to supplied identifier

  404 Not Found – object matched to the criteria is not found, i.e. the current subscriber is not found or undefined

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/whoami"	Return the details for the current active subscriber

- ***`POST`*** ***`https://<Wyde bridge domain>/subscribers/logout`*** – This method performs the logout of the current active subscriber, i.e. clears the web services authorization; the next web services call will require new authorization.

  Returns Status:

  204 No Content – the logout was successfully performed

  Samples:

Call	Description
curl -v -u admin:admin -X POST http://192.168.1.32/wyderef/subscribers/logout	Perform the logout

- ***`POST`*** ***`https://<Wyde bridge domain>/subscribers`*** – This method creates a new Subscriber with the details posted in the data input parameter (pay attention to the list of mandatory fields to be filled in).

Parameters:

`data` – The Subscriber object properties: sid (ignored), details, email, firstName, lastName, login (*), parented (*), password (*), phoneNumber, role (*), meetingRooms (if you would like to create subscriber's meeting rooms simultaneously with the subscriber creation, you should populate meetingRooms property of the subscriber class)

Returns Data:

The URL (location header) to the new created object, e.g. *https://<Wyde bridge domain>/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

201 Created – the object was created; returns the new URL (location header) to the created object

400 Bad Request – the subscriber object was not created, for example if wrong *data* were specified (e.g. wrong field name or field value were specified, mandatory parameter missing, etc.); short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X POST --data "{\"login\":\"testing\",\"password\":\"test\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers	Create new 'testing' subscriber, specify just its *login* and *password* (all other subscriber properties should be omitted to use default values)
curl -v -u admin:admin -X POST --data "{\"firstName\":\"John\",\"lastName\":\"Smith\",\"login\":\"test\",\"phoneNumber\":\"9197473626\",\"password\":\"test\",\"role\":3}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers	Create new 'test' subscriber, specify values for the properties: *login*, *password*, *firstName*, *lastName*, *phoneNumber*, *role*.
curl -v -u admin:admin -X POST --data "{\"firstName\":\"Francis\",\"lastName\":\"Kramer\",\"login\":\"testWithMeetingRooms\",\"password\":\"test\",\"role\":3,\"meetingRooms\":[{\"attributes\":[],\"callFlowId\":1,\"description\":\"\",\"meetingNumber\":887766}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers	Create new 'testWithMeetingRooms' subscriber, specify values for the properties: *login*, *password*, *firstName*, *lastName*, *role*, as well as create the *meeting room* for the subscriber
curl -v -u admin:admin -X POST --data "{\"firstName\":\"Julie\",\"lastName\":\"Kraft\",\"login\":\"testWithMeetingKeys\",\"password\":\"test\",\"role\":3,\"meetingRooms\":[{\"callFlowId\":1,\"description\":\"\",\"keychain\":[{\"accessCode\":\"296048\",\"didGroupId\":5,\"role\":1}],\"meetingNumber\":887766}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers	Create new 'testWithMeetingKeys' subscriber, specify values for the properties: *login*, *password*, *firstName*, *lastName*, *role*, as well as create the *meeting room* for the subscriber and define the host *meeting key* for it

curl -v -u admin:admin -X POST --data "{\"firstName\":\"Matt\",\"lastName\":\"Morrison\",\"login\":\"testWithAttributes\",\"password\":\"test\",\"role\":3,\"meetingRooms\":[{\"callFlowId\":1,\"description\":\"\",\"attributes\":[{\"name\":\"conference_entrytones\",\"value\":\"on\"}],\"meetingNumber\":887766}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers	Create new 'testWithAttributes' subscriber, specify values for the properties: *login*, *password*, *firstName*, *lastName*, *role*, as well as create the *meeting room* for the subscriber and define an *attribute* for it

Note:

This method replaces `createSubscriber (Subscriber subscriber)` from SOAP API.

- `PUT https://<Wyde bridge domain>/subscribers/{SID}` – This method updates existing subscriber referenced by the security identifier *{SID}* with the new information posted in the data input parameter. Please make sure you filled all information that needs to be in the updated Subscriber. Recommendation is to get the subscriber info first, change some info and then call this method to update the information.

Parameters:

`SID` – The security identifier of the Subscriber that should be updated

`data` – The Subscriber object properties: sid (can be omitted, but if specified must match to *{SID}* from the PUT URL), details, email, firstName, lastName, login (*), parented (*), password (*), phoneNumber, role (*), meetingRooms (if you would like to update subscriber's meeting rooms for the subscriber)

Returns Data:

The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the Subscriber object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the SID from the POST URL and SID from data parameter do not match, wrong (nonexistent) SID was specified, etc.; short error description would be supplied in the plain text format

404 Not Found – object matched to the criteria is not found, i.e. the subscriber with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X PUT --data "{\"sid\":\"S91s93FslPe-y0.6sAW2XMKp8IYqxml7\", \"password\":\"new_password\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7	Update the *password* for the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'* Note: for this sample we specify object *sid* parameter in posted data

Call	Description
curl -v -u admin:admin -X PUT --data "{\"firstName\":\"John\",\"lastName\":\"Smith\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7	Update the *firstName* and the *lastName* fields for the subscriber with identifier *'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'* Note: for this sample we do not specify object *sid* parameter in posted data

Note:

To add and modify subscriber's meeting rooms use the resource: `/subscribers/{SID}/meetingRooms`; to add and modify subscriber's meeting keys use the resource: `subscribers/{SID}/meetingRooms/{meetingNumber}/keychain`; see samples below in *Meeting Rooms Management* and *Meeting Keys Management*. This method replaces **`updateSubscriber`** `(Subscriber subscriber)` from SOAP API.

- `DELETE` **`https://<Wyde bridge domain>/subscribers/{SID}`** – This method deletes specific subscriber referenced by the *{SID}* and all his subordinate meeting rooms from the server.

Parameters:

`SID` – The security identifier of the Subscriber that should be deleted

Returns Status:

204 No Content – the object was successfully deleted
404 Not Found – the Subscriber with specified identifier is not found
500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7	Delete subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*

Note:

This method replaces `deleteSubscriber (long subscriberId)` from SOAP API.

Meeting Rooms Management

- `GET` **`https://<Wyde bridge domain>/meetingRooms`** – This method returns the list of all meeting rooms which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output.

Parameters:

`offset` – long zero based offset in recordset.
`limit` – long maximum number of objects to return.
`filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more meeting rooms field names. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any single meeting rooms field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

`includeFields` – comma separated list fields that should be included in the response (e.g. *meetingNumber,description*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes,keychain*).

Returns Data:

list of MeetingRoom objects

Returns Status:

200 OK – returns list of MeetingRoom objects with the specified fields presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetingRooms"	Return all meeting rooms with all fields
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetingRooms?offset=3&limit=3&order=callFlowId%20desc&includeFields=meetingNumber%2Cdescription%2CcallFlowId"	Return *3* meeting rooms *4th* till *6th*; the objects should be sorted by their *callFlowId* descending; only *meetingNumber*, *description*, and *callFlowId* fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetingRooms?filter=callFlowId%20eq%202&order=meetingNumber&excludeFields=attributes"	Return meeting rooms of the *call flow* 2; the objects should be sorted by the meeting number ascending; no *attributes* fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetingRooms?filter=meetingKey.accessCode%20eq%203181&excludeFields=attributes"	Return meeting rooms that have the meeting key with *access code* equal to 3181; no *attributes* fields should be returned

- *GET* `https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms` – This method returns the list of the meeting rooms assigned to the specific subscriber referenced by the security identifier *{SID}* provided which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output.

  Parameters:

  `SID` – The subscriber security identifier

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more meeting rooms field names. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any single meeting rooms field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

`includeFields` – comma separated list fields that should be included in the response (e.g. *meetingNumber,description*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes,keychain*).

Returns Data:

list of MeetingRoom objects

Returns Status:

200 OK – returns list of MeetingRoom objects with the specified fields presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms"	Return all meeting rooms with all fields for subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms?offset=0&limit=2&order=callFlowId%20desc&includeFields=meetingNumber%2Cdescription%2CcallFlowId"	Return first 2 meeting rooms (i.e. from *1th* till *2th*) for subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*; the objects should be sorted by their *callFlowId* descending; only *meetingNumber*, *description*, and *callFlowId* fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms?filter=callFlowId%20eq%202&order=meetingNumber&excludeFields=attributes"	Return meeting rooms of the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*; the *call flow* should be equal 2; the objects should be sorted by the meeting number ascending; no *attributes* fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms?filter=meetingKey.accessCode%20eq%203181&excludeFields=attributes"	Return meeting rooms of the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*; return only meeting room that has the meeting key with *access code* equal to 3181; no *attributes* fields should be returned

- *GET* `https://<Wyde bridge domain>/meetingRooms/{meetingNumber}` – This method returns the full details about the single meeting room referenced by the meeting number *{meetingNumber}* provided.

  Parameters:

  `meetingNumber` – The meeting number of the meeting room

`includeFields` – comma separated list fields that should be included in the response (e.g. *meetingNumber,description*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes,keychain*).

Returns Data:

MeetingRoom object

Returns Status:

200 OK – returns single MeetingRoom object according to supplied meeting number

404 Not Found – object matched to the criteria is not found, i.e. the meeting room with specified meeting number of the specified subscriber is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetingRooms/314159"	Return the details for the meeting room with the meeting number 314159

- *GET* `https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}` – This method returns the full details about the single meeting room referenced by the meeting number *{meetingNumber}* assigned to the specific subscriber referenced by the security identifier *{SID}* provided.

  Parameters:

  `SID` – The subscriber security identifier

  `meetingNumber` – The meeting number of the meeting room

  `includeFields` – comma separated list fields that should be included in the response (e.g. *meetingNumber,description*).

  `excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes,keychain*).

  Returns Data:

  MeetingRoom object

  Returns Status:

  200 OK – returns single MeetingRoom object according to supplied meeting number and subscriber identifier

  404 Not Found – object matched to the criteria is not found, i.e. the meeting room with specified meeting number of the specified subscriber is not found

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/314159"	Return the details for the meeting room with the meeting number 314159 that assigned to the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*

- *POST* `https://<Wyde bridge domain>/meetingRooms` – This method creates a new meeting room with the details posted in the data input parameter (pay attention to the list of mandatory fields to be filled in).
  Parameters:
  `data` – The MeetingRoom object properties: meetingNumber (pass *-1* or omit this parameter to create new unique meeting number), callFlowId, description, keychain (list of MeetingKey objects), and attributes (list of call flow attributes objects)
  Returns Data:
  The URL (location header) to the new created object, e.g. *https://<Wyde bridge domain>/meetingRooms/223344*; it is also being returned in the *Location* identifier of the response headers
  Returns Status:
  201 Created – the object was created; returns the new URL (location header) to the created object
  400 Bad Request – the MeetingRoom object was not created, for example if wrong *data* were specified (e.g. wrong field name or field value were specified, mandatory parameter missing, etc.); short error description would be supplied in the plain text format
  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable
  Samples:

Call	Description
curl -v -u admin:admin -X POST --data "{\"meetingNumber\":334411,\"primaryDidGroupId\":334, \"subscriberSid\":\"S91s93FslPe-y0.6sAW2XMKp8IYqxml7\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetingRooms	Create new meeting room for the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*; the *meeting number* should be 334411, *primaryDidGroupId* should be 334 (all other meeting room properties should be omitted to use default values)
curl -v -u admin:admin -X POST --data "{\"meetingNumber\":334412,\"primaryDidGroupId\":334, \"subscriberSid\":\"S91s93FslPe-y0.6sAW2XMKp8IYqxml7\", \"attributes\": [{\"name\":\"conference_entrytones\", \"value\": \"on\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetingRooms	Create new meeting room for the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*; the *meeting number* should be 334412, *primaryDidGroupId* should be 334, *call flow attribute* `` `conference_entrytones` `` should be set `` `on` ``

- *POST* `https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms` – This method creates a new meeting room assigned to the specified subscriber with the details posted in the data input parameter (pay attention to the list of mandatory fields to be filled in). Please note that you can create meeting rooms by creating new subscriber and providing list of meeting rooms there.
  Parameters:
  `SID` – The subscriber security identifier
  `data` – The MeetingRoom object properties: meetingNumber (pass *-1* or omit this parameter to create new unique meeting number), callFlowId, description,

keychain (list of MeetingKey objects), and attributes (list of call flow attributes objects)

Returns Data:

The URL (location header) to the new created object, e.g. *https://<Wyde bridge domain>/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334406*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

201 Created – the object was created; returns the new URL (location header) to the created object

400 Bad Request – the MeetingRoom object was not created, for example if wrong *data* were specified (e.g. wrong field name or field value were specified, mandatory parameter missing, etc.); short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X POST --data "{\"meetingNumber\":334401, \"primaryDidGroupId\":334}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms	Create new meeting room for the subscriber with *security identifier* '*S91s93FslPe-y0.6sAW2XMKp8IYqxml7*'; the *meeting number* should be 334401, *primaryDidGroupId* should be 334 (all other meeting room properties should be omitted to use default values)
curl -v -u admin:admin -X POST --data "{\"primaryDidGroupId\":334}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms	Create new meeting room for the subscriber with *security identifier* '*S91s93FslPe-y0.6sAW2XMKp8IYqxml7*'; *primaryDidGroupId* should be 334, omit the *meeting number* to auto-generate it by the bridge (all other meeting room properties also should be omitted to use default values)
curl -v -u admin:admin -X POST --data "{\"meetingNumber\":-1, \"primaryDidGroupId\":334, \"description\":\"testing meeting room\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms	Create new meeting room for the subscriber with *security identifier* '*S91s93FslPe-y0.6sAW2XMKp8IYqxml7*'; *primaryDidGroupId* should be 334, specify the *meeting number* -1 to auto-generate it by the bridge, *description* should be `testing meeting room` (all other meeting room properties also should be omitted to use default values)

curl -v -u admin:admin -X POST --data "{\"meetingNumber\":334402,\"primaryDidGroupId\":334, \"attributes\": [{\"name\":\"conference_entrytones\", \"value\": \"on\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms	Create new meeting room for the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*; the *meeting number* should be 334402, *primaryDidGroupId* should be 334, *call flow attribute* `conference_entrytones` should be set `on`
curl -v -u admin:admin -X POST --data "{\"meetingNumber\":334403,\"primaryDidGroupId\":334, \"attributes\": [{\"name\":\"conference_entrytones\", \"value\": \"on\"}], \"keychain\": [{\"role\":1, \"accessCode\":\"112233\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms	Create new meeting room for the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*; the *meeting number* should be 334403, *primaryDidGroupId* should be 334, *call flow attribute* `conference_entrytones` should be set `on`, define the meeting key (keychain) for host (*role* equals to 1) with the *accessCode* equals to 112233

- *PUT* `https://<Wyde bridge domain>/meetingRooms/{meetingNumber}` – This method updates existing meeting room referenced by the meeting number *{meetingNumber}* provided with the new information posted in the data input parameter. Please make sure you filled all information that needs to be in the updated meeting room. Recommendation is to get the meeting room info first, change some info and then call this method to update the information.

  Parameters:

  `meetingNumber` – The meeting number of the meeting room that should be updated

  `data` – The MeetingRoom object properties: meetingNumber (can be omitted, but if specified must match to *{meetingNumber}* from the PUT URL), callFlowId, description, keychain (list of MeetingKey objects), and attributes (list of call flow attributes objects)

  Returns Data:

  The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/meetingRooms/223344*; it is also being returned in the *Location* identifier of the response headers

  Returns Status:

  200 OK – the object was updated; returns the URL (location header) to the updated object

  400 Bad Request – the MeetingRoom object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the meeting number from the POST URL and the meeting number from data parameter do not match, wrong (nonexistent) meeting number was specified, etc.; short error description would be supplied in the plain text format

  404 Not Found – object matched to the criteria is not found, i.e. the meeting room with specified meeting number is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X PUT --data "{\"meetingNumber\":334405, \"description\":\"new test\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetingRooms/334405	Update the *description* of the meeting room with the *meeting number* 334405 Note: for this sample we specify the *meetingNumber* parameter in posted data
curl -v -u admin:admin -X PUT --data "{\"attributes\":[{\"name\":\"conference_entrytones\",\"value\": \"off\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetingRooms/334405	Update the *call flow attribute* `conference_entrytones` to `off` value for the meeting room with the *meeting number* 334405 Note: for this sample we do not specify the *meetingNumber* parameter in posted data

- *PUT* ***https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}*** – This method updates existing meeting room referenced by the meeting number *{meetingNumber}* assigned to the specific subscriber referenced by the security identifier *{SID}* provided with the new information posted in the data input parameter. Please make sure you filled all information that needs to be in the updated meeting room. Recommendation is to get the meeting room info first, change some info and then call this method to update the information.

  Parameters:

  `SID` – The security identifier of the Subscriber whose meeting room should be updated

  `meetingNumber` – The meeting number of the meeting room that should be updated

  `data` – The MeetingRoom object properties: meetingNumber (can be omitted, but if specified must match to *{meetingNumber}* from the PUT URL), callFlowId, description, keychain (list of MeetingKey objects), and attributes (list of call flow attributes objects)

  Returns Data:

  The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334406*; it is also being returned in the *Location* identifier of the response headers

  Returns Status:

  200 OK – the object was updated; returns the URL (location header) to the updated object

  400 Bad Request – the MeetingRoom object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the SID from the POST URL and SID from data parameter do not match, wrong (nonexistent) SID was specified, etc.; short error description would be supplied in the plain text format

  404 Not Found – object matched to the criteria is not found, i.e. the meeting room with specified meeting number is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X PUT --data "{\"meetingNumber\":334406, \"description\":\"new test\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334406	Update the *description* of the meeting room with the *meeting number* 334406 for subscriber with the *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'* Note: for this sample we specify the *meetingNumber* parameter in posted data
curl -v -u admin:admin -X PUT --data "{\"attributes\":[{\"name\":\"conference_entrytones\",\"value\": \"off\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334406	Update the *call flow attribute* `conference_entrytones` to `off` value for the meeting room with the *meeting number* 334406 for subscriber with the *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'* Note: for this sample we do not specify the *meetingNumber* parameter in posted data

- *DELETE* ***https://<Wyde bridge domain>/meetingRooms/{meetingNumber}*** – This method deletes specific meeting room referenced by the meeting number *{meetingNumber}* and all its subordinate meeting keys from the server.

  Parameters:

  `meetingNumber` – The meeting number of the meeting room that should be deleted

  Returns Status:

  204 No Content – the object was successfully deleted

  404 Not Found – the MeetingRoom with specified meeting number is not found

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/meetingRooms/614287	Delete the meeting room with the *meeting number* equals to 614287

- *DELETE* ***https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}*** – This method deletes specific meeting room referenced by the meeting number *{meetingNumber}* assigned to the specific subscriber referenced by the security identifier *{SID}* and all its subordinate meeting keys from the server.

  Parameters:

  `SID` – The security identifier of the Subscriber whose meeting room should be deleted

  `meetingNumber` – The meeting number of the meeting room that should be deleted

  Returns Status:

  204 No Content – the object was successfully deleted

  404 Not Found – the MeetingRoom with specified meeting number is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/532280	Delete the meeting room with the *meeting number* equals to 532280 for subscriber with the *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*

Meeting Keys Management

- `GET https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}/keychain` – This method returns the list of the meeting keys of the specific meeting room referenced by the meeting number *{meetingNumber}* assigned to the specific subscriber referenced by the security identifier *{SID}* provided which match the *filter* provided according to the specified *order*. There are rare cases when this method needs to be called directly as getting subscribers and getting subscriber's meeting rooms returns list of subordinate meeting keys.

  Parameters:

  `SID` – The subscriber security identifier

  `meetingNumber` – The meeting number of the subscriber's meeting room

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more meeting rooms field names. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

  `order` – string specifying any single meeting rooms field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

  Returns Data:

  list of MeetingKey objects

  Returns Status:

  200 OK – returns list of MeetingKey objects presented according to supplied range, filter and order

  400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain"	Return all meeting keys for the meeting room with meeting number *334401* assigned to the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*
curl -v -u admin:admin " http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain?filter=role%20eq%201&order=accessCode"	Return *hosts* (role = 1) meeting keys for the meeting room with meeting number *334401* assigned to the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'* ordered by *accessCode* ascending

Note:

This method replaces `getConfusers (long offset, long limit, String filter, String order)` from SOAP API.

- `GET https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}/keychain/{meetingKeyId}` – This method returns the full details about the single meeting key referenced by the identifier *{meetingKeyId}* of the specific meeting room referenced by the meeting number *{meetingNumber}* assigned to the specific subscriber referenced by the security identifier *{SID}* provided.

  Parameters:

  `SID` – The subscriber security identifier

  `meetingNumber` – The meeting number of the subscriber's meeting room

  `meetingKeyId` – The meeting key identifier

  Returns Data:

  MeetingKey object

  Returns Status:

  200 OK – returns single MeetingKey object according to supplied identifier

  404 Not Found – object matched to the criteria is not found, i.e. the meeting key with specified identifier of the specified subscriber and meeting room is not found

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain/108"	Return the details for the meeting key 108 of the meeting room with the *meeting number* 334401 of the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*

  Note:

  This method replaces `getConfuser (long confuserId)` from SOAP API.

- `POST https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}/keychain` – This method creates a new meeting key for the specified meeting room assigned to the specified subscriber with the details posted in the data input parameter (pay attention to the list of mandatory fields to be filled in). Please note that you can create meeting keys

by creating new subscriber and/or creating new meeting room and providing list of meeting keys there.

Parameters:

`SID` – The subscriber security identifier whose meeting key should be created

`meetingNumber` – The meeting key should be created for the meeting room identified by this meeting number

`data` – The MeetingKey object properties: meetingKeyId (ignored), accessCode (omit this parameter to generate new unique random access code), didGroupId, role

Returns Data:

The URL (location header) to the new created object, e.g. *https://<Wyde bridge domain>/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain/108*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

201 Created – the object was created; returns the new URL (location header) to the created object

400 Bad Request – the MeetingKey object was not created, for example if wrong *data* were specified (e.g. wrong field name or field value were specified, mandatory parameter missing, etc.); short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X POST --data "{\"accessCode\":\"3141\", \"didGroupId\":6, \"role\":1}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain	Create new meeting key for host (*role* = 1) with the specified *accessCode* and *didGroupId* for the meeting room with the *meeting number* 334401 assigned to the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*
curl -v -u admin:admin -X POST --data "{\"accessCode\":\"3171\", \"didGroupId\":6, \"role\":2}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain	Create new meeting key for participant (*role* = 2) with the specified *accessCode* and *didGroupId* for the meeting room with the *meeting number* 334401 assigned to the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*

Note:

This method replaces `createConfuser (Confuser confuser)` from SOAP API.

- `PUT https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}/keychain/{meetingKeyId}` – This method updates existing meeting key referenced by its identifier *{meetingKeyId}* of the meeting room referenced by the meeting number *{meetingNumber}* assigned to the specific subscriber referenced by the security identifier *{SID}* provided with the new information posted in the data input parameter. Please make sure you filled all information that needs to be in the updated meeting key. Recommendation is to get the

meeting key info first, change some info and then call this method to update the information.

Parameters:

`SID` – The security identifier of the Subscriber whose meeting key should be updated

`meetingNumber` – The meeting number of the meeting room that owns the meeting key that should be updated

`meetingKeyId` – The identifier of the meeting key that should be updated

`data` – The MeetingKey object properties: meetingKeyId (can be omitted, but if specified must match to *{meetingKeyId}* from the PUT URL), accessCode, didGroupId, role

Returns Data:

The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain/108*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the MeetingRoom object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the SID from the POST URL and SID from data parameter do not match, wrong (nonexistent) SID was specified, etc.; short error description would be supplied in the plain text format

404 Not Found – object matched to the criteria is not found, i.e. the meeting room with specified meeting number is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X PUT --data "{\"accessCode\":\"1122\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain/108	Update the *accessCode* for the for the meeting key with *meetingKeyId* 108 of the meeting room with the *meeting number* 334401 of the subscriber with *security identifier 'S91s93FslPe-y0.6sAW2XMKp8IYqxml7'*

Note:

This method replaces `updateConfuser (Confuser confuser)` from SOAP API.

- `DELETE https://<Wyde bridge domain>/subscribers/{SID}/meetingRooms/{meetingNumber}/keychain/{meetingKeyId}` – This method deletes the specific meeting key referenced by its identifier *{meetingKeyId}* of the meeting room referenced by the meeting number *{meetingNumber}* assigned to the specific subscriber referenced by the security identifier *{SID}* from the server.

Parameters:

`SID` – The security identifier of the Subscriber whose meeting key should be deleted

`meetingNumber` – The meeting number of the meeting room that owns the meeting key that should be deleted

`meetingKeyId` – The identifier of the meeting key that should be deleted

Returns Status:

204 No Content – the object was successfully deleted

404 Not Found – the MeetingKey with specified meeting number is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/subscribers/S91s93FslPe-y0.6sAW2XMKp8IYqxml7/meetingRooms/334401/keychain/109	Delete the meeting key with *identifier* 109 of the meeting room with *meeting number* 334401 assigned to the subscriber with *security identifier* '*S91s93FslPe-y0.6sAW2XMKp8IYqxml7*'

Note:

This method replaces ***`deleteConfuser (long confuserId)`*** from SOAP API.

Meetings and Calls Management

- ***`GET https://<Wyde bridge domain>/meetings`*** – This method returns the list of all started meetings which are registered for the subscriber on which behalf this call is executed according to specified *filter* and *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output. Please note that field *attributes* in the meeting will be populated with the list of call flow attributes for this meeting; if the meeting is operator-meeting the *operatorStatus* field represents the operator's activity.

  Parameters:

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more meeting field names. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

  `order` – string specifying any single meeting field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

  `includeFields` – comma separated list fields that should be included in the response (e.g. *meetingNumber,created*).

  `excludeFields` – comma separated list fields that should be excluded from the response (e.g. *operatorStatus,broadcastingMode* or *attributes*).

  Returns Data:

  list of Meeting objects

Returns Status:

200 OK – returns list of Meeting objects presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetings?offset=20&limit=10"	Return 10 started meetings from 21 till 30
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetings?offset=0&limit=5&filter=meetingNumber%20ge%20100"	Return first 5 started meetings where the *meeting number* grater than 100
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetings?offset=0&limit=5&filter=id%20ge%20100%20and%20holdMode%20ne%200&excludeFields=operatorStatus%2CbroadcastingMode"	Return first 5 started meetings where the *meeting identifier* grater than 100 and *hold mode* not equal 0 (the meeting is on hold); exclude fields: *operatorStatus,broadcastingMode*
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetings?offset=0&limit=5&filter=meetingNumber%20ge%20100%20and%20muteMode%20eq%2032&order=meetingNumber%20desc"	Return first 5 started meetings where the *meeting number* grater than 100 and *mute mode* equal 32; the meetings should be sorted by *meeting number* descending

Note:

This method replaces `getMeetings (long offset, long limit, String filter, String order)` from SOAP API.

- `GET https://<Wyde bridge domain>/meetings/{meetingNumber}` – This method returns the full details about the Meeting referenced by the meeting number *{meetingNumber}* provided.

  Parameters:

  `meetingNumber` – The Meeting number

  `includeFields` – comma separated list fields that should be included in the response (e.g. *meetingNumber,created*).

  `excludeFields` – comma separated list fields that should be excluded from the response (e.g. *operatorStatus,broadcastingMode* or *attributes*).

  Returns Data:

  Meeting object

  Returns Status:

  200 OK – returns single Meeting object according to supplied meeting number

  404 Not Found – object matched to the criteria is not found, i.e. the meeting with specified meeting number is not found

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetings/111"	Return the details for started meetings with the meeting number 111

Note:

This method replaces `getMeeting (long meetingId)` from SOAP API.

- `GET https://<Wyde bridge domain>/meetings/{meetingNumber}/attributes` – This method returns the list of the call flow attributes for the started meeting referenced by the meeting number *{meetingNumber}* which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application. The meeting attributes are either defined for the call flow or they could be overridden for the meeting.

  Parameters:

  `meetingNumber` – The meeting number of the meeting which attributes should be returned

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more callflow's attribute field names. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

  `order` – string specifying any single callflow's attribute field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

  Returns Data:

  list of callflow's Attribute objects

  Returns Status:

  200 OK – returns list of callflow's Attribute objects presented according to supplied range, filter and order

  400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/meetings/111/attributes?order=name"	Return all call flow attributes for the meeting with *meeting number* 111 ordered by *name* of the attribute ascending

- `PUT https://<Wyde bridge domain>/meetings/{meetingNumber}` – This method changes the meeting state of the specific meeting referenced by the meeting number *{meetingNumber}* with the new parameters posted in the *data* input. Thus this method performs the following actions:

- Makes the meeting secured, i.e. moves the meeting referenced by number into the state when no new calls are allowed to get in there:
  - ✓ data: isSecured = true
- Cancels effect of securing meeting, i.e. new calls can join the Meeting:
  - ✓ data: isSecured = false
- Places the meeting on hold; hold mode bitmask flag consisting of 3 groups for *hosts*, *participants*, and *listeners* and each of the group has bits that determines whether the group is *on hold* or *online* (is not on hold) – see details in Table 9 for *holdMode* property:
  - ✓ data: holdMode = 40
- Places the meeting off hold:
  - ✓ data: holdMode = 0
- Mutes all participants (it doesn't touch moderators); mute mode bitmask flag consisting of 3 groups for *hosts*, *participants*, and *listeners* and each of the group has bits that determines the specific mute mode *open* (when all can speak or mute themselves), *relaxed* (when the callers in the role are muted, but they can un-mute themselves), and *strict* (when the callers in the role cannot un-mute themselves) – see details in Table 9 for *muteMode* property:
  - ✓ data: muteMode = 32 (the listeners are muted and cannot un-mute themselves, participants and hosts are not muted)
  - ✓ data: muteMode = 36 (the listeners are muted and cannot un-mute themselves, participants are muted and can un-mute themselves, and hosts are not muted)
  - ✓ data: muteMode = 40 (the listeners and participants are muted and cannot un-mute themselves, hosts are not muted)
- Starts, stops or clears Q&A queue for the specific meeting (if Q&A is enabled participants can put themselves into the question queue so moderator can pick a questioner:
  - ✓ data: qaMode = 1 (starts Q&A mode for the meeting)
  - ✓ data: qaMode = 2 (talk to the next caller in Q&A queue for the meeting)
  - ✓ data: qaMode = 4 (clears Q&A queue for the meeting)
  - ✓ data: qaMode = 0 (stops Q&A mode for the meeting)
- Starts the meeting recording; if the meeting call flow is CONF, and the recording method (*recording_method*) call flow attribute value is either "local+pin" or "remote+pin", the subscriber's pin (that usually could be requested from the user) and the meeting host access code should be transferred to this method as its parameters *pin* and *accessCode* to perform recording authorization; otherwise these parameters should be empty (omitted);
  - ✓ data: isRecording = true
- Stops the meeting recording:
  - ✓ data: isRecording = false
- Starts the polling within specific meeting with selected options (the same as default #5 on touch tone keypad) – see details in Table 9 for *pollingMode* property:
  - ✓ data: *pollingMode* = "{selected polling options}" (digits 1, 2, ..., 9, 0)
- Stops the polling within specific meeting:
  - ✓ data: *pollingMode* = ""

Parameters:

`meetingNumber` – The meeting number of the meeting that should be updated, i.e. the meeting which state should be amended

`data` – The Meeting object properties; see above to determine what meeting properties should be changed and what properties' values should be set to change the meeting state

Returns Data:

The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/meetings/49843764*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the Meeting object was not updated, i.e. the meeting state was not changed, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the ID from the POST URL and ID from data parameter do not match, wrong (nonexistent) ID was specified, etc.; short error description would be supplied in the plain text format

404 Not Found – object matched to the criteria is not found, i.e. the meeting with specified meeting number is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X PUT --data "{\"isSecured\":\"true\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Make the meeting with the *meeting number* 111 secured
curl -v -u admin:admin -X PUT --data "{\"isSecured\":\"false\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Make the meeting with the *meeting number* 111 unsecured
curl -v -u admin:admin -X PUT --data "{\"holdMode\":40}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Put the meeting with the *meeting number* 111 on hold, i.e. put all listeners and participants on hold
curl -v -u admin:admin -X PUT --data "{\"holdMode\":0}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/49843764	Take the meeting with the *meeting number* 111 off hold
curl -v -u admin:admin -X PUT --data "{\"muteMode\":32}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Set the meeting with the *meeting number* 111 mute mode when the listeners are muted and cannot un-mute themselves, participants and hosts are not muted (muteMode = 32)

Call	Description
curl -v -u admin:admin -X PUT --data "{\"muteMode\":36}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Set the meeting with the *meeting number* 111 mute mode when the listeners are muted and cannot un-mute themselves, participants are muted and can un-mute themselves, and hosts are not muted (muteMode = 36, i.e. *relaxed*)
curl -v -u admin:admin -X PUT --data "{\"isRecording\":\"true\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Start the recording of the meeting with the *meeting number* 111
curl -v -u admin:admin -X PUT --data "{\"isRecording\":\"false\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Stop the recording of the meeting with the *meeting number* 111
curl -v -u admin:admin -X PUT --data "{\"qaMode\":1}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Start *Q&A mode* for the meeting with the *meeting number* 111
curl -v -u admin:admin -X PUT --data "{\"qaMode\":2}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Talk to the next caller in Q&A queue for the meeting with the *meeting number* 111 that is in *Q&A mode*
curl -v -u admin:admin -X PUT --data "{\"qaMode\":4}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Clear Q&A queue for the meeting with the *meeting number* 111 that is in *Q&A mode*
curl -v -u admin:admin -X PUT --data "{\"qaMode\":0}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/meetings/111	Stop *Q&A mode* for the meeting with the *meeting number* 111

Note:

This method replaces:

```
secureConference (long conferenceId),
unSecureConference (long conferenceId),
holdConference (long conferenceId),
unHoldConference (long conferenceId),
muteConference (long conferenceId, long mode),
qaSetMode (long conferenceId, long mode),
qaMuteMode (long conferenceId, long mode),
startConferenceRecording (long conferenceId, String pin,
    String accessCode),
stopConferenceRecording (long conferenceId),
startPolling (long conferenceId, String keys),
stopPolling (long conferenceId)
```

from SOAP API.

- `DELETE https://<Wyde bridge domain>/meetings/{meetingNumber}` – This method causes all calls to be dropped from the meeting and the meeting to be terminated (hanged up); the meeting is referenced by the meeting number *{meetingNumber}* provided.

  Parameters:

  `meetingNumber` – The meeting number of the meeting that should be hanged up

Returns Status:

204 No Content – the object was successfully deleted, i.e. the meeting was terminated (hanged up)

404 Not Found – the meeting with specified meeting number is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/meetings/111	Drop the meeting with *meeting number* 111

Note:

This method replaces `hangupConference (long conferenceId)` from SOAP API.

- `GET https://<Wyde bridge domain>/calls` – This method returns the list of all started calls which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application. If this method is called from non admin Subscribers it will returns only Calls visible for this account; if call doesn't present an access code yet – it is visible only by admin.

Parameters:

`offset` – long zero based offset in recordset.

`limit` – long maximum number of objects to return.

`filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more call field names (any call fields are permitted). Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any single call field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

Returns Data:

list of Call objects

Returns Status:

200 OK – returns list of Call objects presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/calls?offset=20&limit=10"	Return 10 started calls from 21 till 30
curl -v -u admin:admin "http://192.168.1.32/wyderef/calls?offset=0&limit=5&filter=id%20ge%20100"	Return first 5 started calls where the call *identifier* grater than 100

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/calls?offset=0&limit=10&filter=meetingNumber%20eq%20111"	Return first 10 started calls where the *meeting number* 111
curl -v -u admin:admin "http://192.168.1.32/wyderef/calls?offset=0&limit=10&filter=meetingNumber%20eq%20111%20and%20role%20eq%202"	Return first 10 participants (i.e. with role `MODE_PARTICIPANT = 2L`) where the *meeting number* 111
curl -v -u admin:admin "http://192.168.1.32/wyderef/calls?offset=0&limit=5&filter=id%20ge%20100%20and%20holdMode%20ne%200"	Return first 5 started calls where the call *identifier* grater than 100 and *hold mode* not equal 0 (the call is on hold)
curl -v -u admin:admin "http://192.168.1.32/wyderef/calls?offset=0&limit=5&filter=id%20ge%20100%20and%20muteMode%20ne%200&order=id%20desc"	Return first 5 started calls where the call *identifier* grater than 100 and *mute mode* not equal 0 (the call is muted); the calls should be sorted by call *identifier* descending

Note:

This method replaces `getCalls (long offset, long limit, String filter, String order)` from SOAP API.

- `GET https://<Wyde bridge domain>/calls/{ID}` – This method returns the full details about the call referenced by the call identifier *{ID}* provided.

Parameters:

`ID` – The Call identifier

Returns Data:

Call object

Returns Status:

200 OK – returns single Call object according to supplied call identifier

404 Not Found – object matched to the criteria is not found, i.e. the call with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/calls/174"	Return the details for started calls with the call *identifier* 174

Note:

This method replaces `getCall (long callId)` from SOAP API.

- `PUT https://<Wyde bridge domain>/calls/{ID}` – This method changes the call state of the specific call referenced by its identifier *{ID}* with the new parameters posted in the *data* input. Thus this method performs the following actions:
  - Places the call on hold; hold mode is flag that determines whether the call is put *on hold* (by administrator, owner, client, etc.) or the call is *online* (is not on hold): *0* value of this flag determines that the call is *online*, *1* value of this flag determines that the call is *on hold* – see details in Table 11 for *holdMode* property:
    - ✓ data: holdMode = 1
  - Places the call off hold (online):
    - ✓ data: holdMode = 0

- Mutes the specific call; mute mode bitmask flag consisting of 2 groups (bits) for *host muting* and *self muting*; and each of the group has bit that determines whether this call is muted or not: *0* value of this flag determines that the call is *not muted* (*host* or *self*) and *1* value of this flag determines that the call is *muted* (*host* or *self*) – see details in Table 11 for *muteMode* property:
  - ✓ data: muteMode = 1 (to mute the call by host)
- Un-mutes the call:
  - ✓ data: muteMode = 0
- Sets the custom name of the caller:
  - ✓ data: customName = "{new custom name to be set}"
- Attaches (moves) the caller to the sub-meeting or detaches it from the sub-meeting; if the *subconference* property value is not empty, the call is being attached to sub-meeting (if it is not currently connected to any of the sub-meetings) or moved to sub-meeting (if it is currently connected to another sub-meeting); non-empty parameter represents the name of the sub-meeting up to 16 characters length (only letters and digits are allowed as the name of the sub-meeting); if the parameter is empty string the call is being detached from the sub-meeting:
  - ✓ data: subconference = {sub-meeting to move the call}
- Sets the specified job code for the specific call:
  - ✓ data: jobCode = {new active job code to be set}
- Changes the role of the call in the meeting (note: no additional CDR record is being created for the call); the role (mode) that will be granted to the call in the meeting could be `MODE_HOST = 1L`, `MODE_PARTICIPANT = 2L`, `MODE_LISTENER = 3L` – see details in Table 11 for *role* property:
  - ✓ data: role = {new role}

Parameters:

`ID` – The identifier of the call that should be updated, i.e. the call which state should be amended

`data` – The Call object properties; see above to determine what call properties should be changed and what properties' values should be set to change the call state

Returns Data:

The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/calls/18274934*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the Call object was not updated, i.e. the call state was not changed, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the ID from the POST URL and ID from data parameter do not match, wrong (nonexistent) ID was specified, etc.; short error description would be supplied in the plain text format

404 Not Found – object matched to the criteria is not found, i.e. the call with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X PUT --data "{\"holdMode\":1}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/calls/179	Put the call with the *identifier* 179 on hold
curl -v -u admin:admin -X PUT --data "{\"holdMode\":0}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/calls/179	Take the call with the *identifier* 179 off hold
curl -v -u admin:admin -X PUT --data "{\"muteMode\":1}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/calls/179	Mute the call with the *identifier* 179
curl -v -u admin:admin -X PUT --data "{\"muteMode\":0}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/calls/179	Un-mute the call with the *identifier* 179
curl -v -u admin:admin -X PUT --data "{\"customName\":\"testing\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/calls/179	Set *custom name* for the caller with the call *identifier* 179 to "testing"
curl -v -u admin:admin -X PUT --data "{\"jobCode\":\"1218\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/calls/179	Set *new job* code "1218" for the call with the *identifier* 179

Note:

This method replaces:

`holdSession (long sessionId),`
`unHoldSession (long sessionId),`
`muteSession (long sessionId),`
`unMuteSession (long sessionId),`
`setCustomName (long sessionId, String name),`
`setSubconference (long[] sessionIds, String subconference),`
`setSessionRole (long sessionId, long role)`
from SOAP API.

- `DELETE https://<Wyde bridge domain>/calls/{ID}` – This method disconnects (hangs up) the call reference by the *{ID}* provided.

Parameters:

`ID` – The identifier of the call that should be disconnected (hanged up)

Returns Status:

204 No Content – the object was successfully deleted, i.e. the call was terminated (hanged up)

404 Not Found – the call with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/wyderef/calls/174	Drop (disconnect) the call with *identifier* 174

Note:

This method replaces `hangupSession (long sessionId)` from SOAP API.

MDRs and CDRs Management

- `GET https://<Wyde bridge domain>/mdrs` – This method returns the list of completed (past) meetings which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application. For non-admin (non-operator) users the method returns meetings that were registered for the subscriber; for administrator and operator it returns whole list of completed meetings. Due the large number of completed meetings, use filtering and paging to avoid slow execution of this method.

  Parameters:

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more Mdr field names (any Mdr fields are permitted; you can also use *createdStr* and *droppedStr* aliases to filter by the string `yyyy.MM.dd/HH:mm:ss` representation of the meeting created and dropped dates). Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

  `order` – string specifying any Mdr field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

  Returns Data:

  list of Mdr objects

  Returns Status:

  200 OK – returns list of Mdr objects according to supplied range, filter and order

  400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/mdrs?offset=0&limit=10&order=created"	Return first 10 completed meetings ordered by the *created* ascending (i.e. date and time when the meeting was started)
curl -v -u admin:admin "http://192.168.1.32/wyderef/mdrs?offset=0&limit=10&filter=meetingNumber%20eq%20223344&order=participantCount%20desc"	Return first 10 completed meetings where *meetingNumber* equals to 223344 ordered by the *participantCount* descending

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/mdrs?offset=0&limit=20&filter=createdStr%20like%20'2014.12.22%25'&order=created"	Return first 20 completed meetings *created* on December 22nd, 2014 ordered by the meeting *created* date ascending
curl -v -u admin:admin "http://192.168.1.32/wyderef/mdrs?offset=0&limit=20&filter=droppedStr%20like%202015.02.%25&order=created"	Return first 20 completed meetings *dropped* in February 2015 ordered by the meeting *created* date ascending

- *PUT* `https://<Wyde bridge domain>/mdrs/{meetingId}` – This method updates existing completed (past) meeting referenced by its identifier *{meetingId}*. Please make sure you filled all information that needs to be in the updated Mdr object.

  Parameters:

  `meetingId` – Unique meeting identifier assigned by server

  `data` – The Mdrs object properties: duration, created, meetingNumber, participantCount, audioParticipantCount, screenSharingParticipantCount, controlParticipantcount, recordingParticipantCount, recordingDuration, webRecordingDiration, videoParticipantCount, description, referenceNumber, jobCode, password, hasRecording, meetingId (ignored), processingStatus(ignored), fileSize (available '0' value for removing recordings, another values ignored), hasPollingResults (ignored), recordingUrl (ignored), sharedRecordingUrl (ignored), expirePeriod (ignored), customExtensions (ignored)

  Returns Data:

  The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/mdrs/134217776*

  Returns Status:

  200 OK – the object was updated; returns the URL (location header) to the updated object

  400 Bad Request – the Mdr object was not updated, for example if wrong *data* were specified, e.g. removing recording in meeting where recordings does not exist

  404 Not Found – object matched to the criteria is not found, i.e. the Mdr object with specified meeting identifier is not found

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin -X PUT --data "{\"password\":\"test\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/mdrs/134217776	Update the *password* for the for the Mdr object with *meetingId* 134217776 of the meeting

- *GET* `https://<Wyde bridge domain>/cdrs` – This method returns the list of completed (past) calls which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application. For non-admin (non-operator) users the method returns calls that were registered for the subscriber; for administrator and operator it returns whole list of

completed calls. Due the large number of completed calls, use filtering and paging to avoid slow execution of this method.

Parameters:

`offset` – long zero based offset in recordset.

`limit` – long maximum number of objects to return.

`filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more Cdr field names (any Cdr fields are permitted; you can also use *createdStr* and *droppedStr* aliases to filter by the string `yyyy.MM.dd/HH:mm:ss` representation of the call created and dropped dates). Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any Cdr field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

Returns Data:

list of Cdr objects

Returns Status:

200 OK – returns list of Cdr objects according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/cdrs?offset=0&limit=10&order=created"	Return first 10 completed calls ordered by the *created* ascending (i.e. date and time when the call was started)
curl -v -u admin:admin "http://192.168.1.32/wyderef/cdrs?offset=0&limit=10&filter=meetingNumber%20eq%20223344&order=meetingNumber%20desc"	Return first 10 completed calls where *meetingNumber* equals to 223344 ordered by the *meeting number* descending
curl -v -u admin:admin "http://192.168.1.32/wyderef/cdrs?offset=0&limit=20&filter=createdStr%20like%20'2014.12.22%25'&order=created"	Return first 20 completed calls *created* on December 22nd, 2014 ordered by the call *created* date ascending
curl -v -u admin:admin "http://192.168.1.32/wyderef/cdrs?offset=0&limit=20&filter=droppedStr%20like%202015.02.%25&order=created"	Return first 20 completed calls *dropped* in February 2015 ordered by the call *created* date ascending

- *PUT* `https://<Wyde bridge domain>/cdrs/{callId}` – This method updates existing completed (past) call referenced by its identifier *{callId}*. Please make sure you filled all information that needs to be in the updated Cdr object.

  Parameters:

  `callId` – Unique call identifier assigned by server

  `data` – The Cdrs object property: customName

  Returns Data:

  The URL (location header) to the updated object, e.g.

  https://<Wyde bridge domain>/cdrs/16777272

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the Cdr object was not updated, for example if wrong *data* were specified, e.g. updating of any properties except the 'customName' property

404 Not Found – object matched to the criteria is not found, i.e. the Cdr object with specified call identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X PUT --data "{\"customName\":\"testing\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/cdrs/16777272	Update the *customName* for the for the Cdr object with *callId* 16777272 of the meeting

Event Logger Information

- `GET https://<Wyde bridge domain>/events` – This method returns the list of logged events which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application. For non-admin (non-operator) users the method returns events that were occurred for the subscriber's meetings; for administrator and operator it returns whole list of occurred events. Due the large number of events, use filtering and paging to avoid slow execution of this method.

  Parameters:

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more event field names (any event fields are permitted). Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

  `order` – string specifying any event field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

  Returns Data:

  list of Event objects

  Returns Status:

  200 OK – returns list of Event objects according to supplied range, filter and order

  400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/events?offset=0&limit=10&order=created"	Return first 10 event records ordered by the *created* ascending (i.e. date and time when the event occurred)

Call Flow, DID and DID Group Management

- `GET https://<Wyde bridge domain>/callflows` – This method returns the list of call flows which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output.

  Parameters:

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more call flow field names (call flow fields *id*, *name*, *path* are permitted). Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

  `order` – string specifying any single call flow field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

  `includeFields` – comma separated list fields that should be included in the response (e.g. *id,name,path*).

  `excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributeTemplates*).

  Returns Data:

  list of CallFlow objects

  Returns Status:

  200 OK – returns list of CallFlow objects with the specified fields presented according to supplied range, filter and order

  400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/callflows"	Return all call flows with their attributes
curl -v -u admin:admin "http://192.168.1.32/wyderef/callflows?offset=2&limit=2&order=name%20desc&excludeFields=attributeTemplates"	Return 2 call flows from 3rd till 4th; the call flows should be sorted by their *name* descending, *attributeTemplates* field should be excluded from the response

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/callflows?filter=id%20eq%202"	Return the call flow with *identifier* equal to 2, all fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/callflows?filter=name%20eq%20SPECTEL"	Return the call flow with SPECTEL *name*, all fields should be returned

Note:

This method replaces `getCallFlows (long offset, long limit, String filter, String order)` from SOAP API.

- `GET https://<Wyde bridge domain>/callflows/{ID}` – This method returns the full details about the call flow referenced by the *{ID}* provided.

Parameters:

`ID` – The CallFlow identifier

`includeFields` – comma separated list fields that should be included in the response (e.g. *id,name,path*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributeTemplates*).

Returns Data:

CallFlaw object

Returns Status:

200 OK – returns single CallFlow object according to supplied identifier

404 Not Found – object matched to the criteria is not found, i.e. the call flow with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/callflows/2"	Return the details for the call flow with *identifier* 2

Note:

This method replaces `getCallFlow (long callFlowId)` from SOAP API.

- `GET https://<Wyde bridge domain>/dids` – This method returns the list of DIDs (phone numbers) registered on the bridge which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output.

Parameters:

`offset` – long zero based offset in recordset.

`limit` – long maximum number of objects to return.

`filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more DID field names. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any single DID field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

`includeFields` – comma separated list fields that should be included in the response (e.g. *phoneNumber,description*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes*).

Returns Data:

list of Did objects

Returns Status:

200 OK – returns list of Did objects with the specified fields presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin "http://192.168.1.32/wyderef/dids"	Return all DID numbers with all their fields
curl -v -u admin:admin "http://192.168.1.32/wyderef/dids?offset=3&limit=3&order=phoneNumber%20desc&excludeFields=attributes"	Return 3 DID numbers from 4th till 6th; the DIDs should be sorted by their *phone numbers* descending; *attributes* field should be excluded from the output
curl -v -u admin:admin "http://192.168.1.32/wyderef/dids?filter=id%20eq%204&includeFields=phoneNumber%2Cdescription%2CdidGroupId"	Return the DID number with *identifier* equal to 4; only *phoneNumber*, *description*, *didGroupId* fields should be returned
curl -v -u admin:admin "http://192.168.1.32/wyderef/dids?filter=phoneNumber%20eq%208665080012"	Return the DID number with *phone number* equal to 8665080012; all DID fields should be returned

Note:

This method replaces `getDNISes (long offset, long limit, String filter, String order)` from SOAP API.

- `GET https://<Wyde bridge domain>/dids/{ID}` – This method returns the full details about the DID referenced by the *{ID}* provided.

Parameters:

`ID` – The DID identifier

`includeFields` – comma separated list fields that should be included in the response (e.g. *phoneNumber,description*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes*).

Returns Data:

Did object

Returns Status:

200 OK – returns single Did object according to supplied identifier

404 Not Found – object matched to the criteria is not found, i.e. the call flow with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/dids/4"	Return the details for the DID with identifier 4

Note:

This method replaces `getDNIS (long dnisId)` from SOAP API.

- ***POST*** ***https://<Wyde bridge domain>/dids*** – This method creates a new DID with the details posted in the data input parameter. Please note that only administrator can create new DIDs.

  Parameters:

  `data` – The Did object properties: id (ignored), didGroupId (*), phoneNumber (*), description, state, attributes

  Returns Data:

  The URL (location header) to the new created object, e.g. *https://<Wyde bridge domain>/dids/19*; it is also being returned in the *Location* identifier of the response headers

  Returns Status:

  201 Created – the object was created; returns the new URL (location header) to the created object

  400 Bad Request – the DID object was not created, for example if wrong *data* were specified (e.g. wrong field name or field value were specified, mandatory parameter missing, etc.); short error description would be supplied in the plain text format

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin -X POST --data "{\"didGroupId\":7,\"description\":\"test did\",\"phoneNumber\":\"8665080013\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/dids	Create DID for Did Group 7 (*didGroupId*=7), *phone number* 8665080013, and the *description* "test did"
curl -v -u admin:admin -X POST --data "{\"didGroupId\":2,\"phoneNumber\":\"8665080011\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/dids	Create DID for Did Group 2 (*didGroupId*=2), *phone number* 8665080011; omit other parameters
curl -v -u admin:admin -X POST --data "{\"didGroupId\":23,\"phoneNumber\":\"8665080018\",\"attributes\":[{\"name\":\"conference_entrytones\",\"value\":\"on\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/dids	Create DID for Did Group 23 (*didGroupId*=23), *phone number* 8665080018; define *call flow attribute* `conference_entrytones` equal `on`

  Note:

  This method replaces `createDNIS (DNIS dnis)` from SOAP API.

- ***PUT*** ***https://<Wyde bridge domain>/dids/{ID}*** – This method updates existing DID referenced by the identifier *{ID}* with the new information posted in the data input parameter. Please note that only administrator has a permission to update DID.

Parameters:

`ID` – The identifier of the DID that should be updated

`data` – The DID object properties: id (can be omitted, but if specified must match to *{ID}* from the PUT URL), didGroupId (*), phoneNumber (*), description, state, attributes (if you would like to update call flow attributes for the DID)

Returns Data:

The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/dids/10*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the Did object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the ID from the POST URL and ID from data parameter do not match, wrong (nonexistent) ID was specified, etc.; short error description would be supplied in the plain text format

404 Not Found – object matched to the criteria is not found, i.e. the DID with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X PUT --data "{\"id\":10,\"*didGroupId*\":1,\"description\":\"testing\",\"phoneNumber\":\"8665080111\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/dids/10	Update DID with *identifier* 10 and change its properties: Did Group to 1 (*didGroupId*=1), *phone number* to 8665080111; and the *description* to "testing"
curl -v -u admin:admin -X PUT --data "{\"*didGroupId*\":7,\"phoneNumber\":\"8665080123\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/dids/10	Update DID with *identifier* 10 and change its properties: Did Group to 7 (*didGroupId*=7), *phone number* to 8665080123; and omit other parameters
curl -v -u admin:admin -X PUT --data "{\"description\":\"new test\",\"phoneNumber\":\"8665080118\",\"attributes\":[{\"name\":\"dnis_language_all\",\"value\":\"en\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/dids/10	Update DID with *identifier* 10 and change its properties: *description* to `new test`, *phone number* to 8665080118; define *call flow attribute* `dnis_language_all` equal `en`

Note:

This method replaces ***`updateDNIS`*** `(DNIS dnis)` from SOAP API.

- ***`DELETE`*** ***`https://<Wyde bridge domain>/dids/{ID}`*** – This method deletes specific DID referenced by the *{ID}* from the server. Please note that only administrator has a permission to delete DID.

Parameters:

`ID` – The identifier of the DID that should be deleted

Returns Status:

204 No Content – the object was successfully deleted

404 Not Found – the DID with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/dids/10	Delete DID with *identifier* 10

Note:

This method replaces `deleteDNIS (long dnisId)` from SOAP API.

- `GET https://<Wyde bridge domain>/didGroups` – This method returns the list of DID groups registered on the bridge which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output.

Parameters:

`offset` – long zero based offset in recordset.

`limit` – long maximum number of objects to return.

`filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more DidGroup field names. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any single DidGroup field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

`includeFields` – comma separated list fields that should be included in the response (e.g. *phoneNumber,description*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes*).

Returns Data:

list of DidGroup objects

Returns Status:

200 OK – returns list of DidGroup objects with the specified fields presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/didGroups"	Return all DID groups
curl -v -u admin:admin "http://192.168.1.32/wyderef/didGroups?offset=3&limit=3&order=description%20desc"	Return 3 DID groups from 4th till 6th; the DIDs groups should be sorted by their *description* descending
curl -v -u admin:admin "http://192.168.1.32/wyderef/didGroups?filter=id%20eq%206"	Return the DID groups with *identifier* equal to 6

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/didGroups?filter=callFlowId%20eq%202&order=description"	Return the DID groups with *call flow identifier* equal to 2; the DIDs groups should be sorted by their *description* ascending

- `GET https://<Wyde bridge domain>/didGroups/{ID}` – This method returns the full details about the DID group referenced by the *{ID}* provided.

  Parameters:

  `ID` – The DID Group identifier

  `includeFields` – comma separated list fields that should be included in the response (e.g. *phoneNumber,description*).

  `excludeFields` – comma separated list fields that should be excluded from the response (e.g. *attributes*).

  Returns Data:

  DidGroup object

  Returns Status:

  200 OK – returns single DidGroup object according to supplied identifier

  404 Not Found – object matched to the criteria is not found, i.e. the call flow with specified identifier is not found

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/didGroups/6"	Return the details for the DID group with *identifier* 6

- `POST https://<Wyde bridge domain>/didGroups` – This method creates a new DID group with the details posted in the data input parameter. Please note that only administrator can create new DID group.

  Parameters:

  `data` – The DidGroup object properties: id (ignored), name (*), callFlowId (*), description, state, dids (if you would like to create DIDs simultaneously with the Did group creation, you should populate dids property of the DidGroup class)

  Returns Data:

  The URL (location header) to the new created object, e.g. *https://<Wyde bridge domain>/didGroups/12*; it is also being returned in the *Location* identifier of the response headers

  Returns Status:

  201 Created – the object was created; returns the new URL (location header) to the created object

  400 Bad Request – the DidGroup object was not created, for example if wrong *data* were specified (e.g. wrong field name or field value were specified, mandatory parameter missing, etc.); short error description would be supplied in the plain text format

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin -X POST --data "{\"name\":\"Testing\",\"callFlowId\":2,\"description\":\"New did group for testing\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/didGroups	Create DID group for call flow 2 (*callFlowId*=2), *name* equal "Testing", and the *description* "New did group for testing"
curl -v -u admin:admin -X POST --data "{\"name\":\"Testing2\",\"callFlowId\":2,\"dids\":[{\"description\":\"test did\",\"phoneNumber\":\"8665080113\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/didGroups	Create DID group for call flow 2 (*callFlowId*=2), *name* equal "Testing2", and at the same time create the DID (*phone number* 8665080113)

- *PUT* `https://<Wyde bridge domain>/didGroups/{ID}` – This method updates existing DID group referenced by the identifier *{ID}* with the new information posted in the data input parameter. Please note that only administrator has a permission to update DID group.

  Parameters:

  `ID` – The identifier of the DID group that should be updated

  `data` – The DidGroup object properties: id (can be omitted, but if specified must match to *{ID}* from the PUT URL), name (*), callFlowId (*), description, state, dids (if you would like to update DIDs for the DID group)

  Returns Data:

  The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/didGroups/12*; it is also being returned in the *Location* identifier of the response headers

  Returns Status:

  200 OK – the object was updated; returns the URL (location header) to the updated object

  400 Bad Request – the DidGroup object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the ID from the POST URL and ID from data parameter do not match, wrong (nonexistent) ID was specified, etc.; short error description would be supplied in the plain text format

  404 Not Found – object matched to the criteria is not found, i.e. the DID with specified identifier is not found

  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

  Samples:

Call	Description
curl -v -u admin:admin -X PUT --data "{\"name\":\"Testing3\",\"description\":\"new testing\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/didGroups/12	Update DID group with *identifier* 12 and change its properties: *name* to "Testing"; and the *description* to "new testing"
curl -v -u admin:admin -X PUT --data "{"dids":[{"description":"test did","phoneNumber":"8665080013"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/didGroups/12	Update DID group with *identifier* 12 and define the DID (*phone number* 8665080013, *description* "test did") for it; and omit other parameters

- *DELETE https://<Wyde bridge domain>/**didGroups/{ID}*** – This method deletes specific DID group referenced by the *{ID}* from the server. Please note that only administrator has a permission to delete DID group.
  Parameters:
  `ID` – The identifier of the DID group that should be deleted
  Returns Status:
  204 No Content – the object was successfully deleted
  404 Not Found – the DID group with specified identifier is not found
  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable
  Samples:

Call	Description
curl -v -u admin:admin -X DELETE http://192.168.1.32/wyderef/didGroups/12	Delete DID group with *identifier* 12

Bridges and Settings Management

- *GET https://<Wyde bridge domain>/**bridge*** – This method returns the details about the current bridge.
  Returns Data:
  Bridge object
  Returns Status:
  200 OK – returns single Bridge object
  404 Not Found – object matched to the criteria is not found, i.e. the current bridge is not found or undefined
  500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable
  Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/bridge"	Return the details for the current bridge

- *GET https://<Wyde bridge domain>/**bridges*** – This method returns the list of bridges which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application; there are also two parameters *includeFields* and *excludeFields* defining what fields should be included or excluded from the output.
  Parameters:
  `offset` – long zero based offset in recordset.
  `limit` – long maximum number of objects to return.
  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more bridge field names (any bridge fields are permitted); several filters can be combined together; example: *role eq 0 and neme ilike '%DEV%'*. Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any single bridge field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

`includeFields` – comma separated list fields that should be included in the response (e.g. *name,ipAddress*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *settings*).

Returns Data:

list of Bridge objects

Returns Status:

200 OK – returns list of Bridge objects with the specified fields presented according to supplied range, filter and order

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/bridges"	Return all bridges with all fields
curl -v -u admin:admin "http://192.168.1.32/wyderef/bridges?filter=id%20eq%201 &includeFields=name%2CipAddress%2Csettings"	Return bridge with *identifier* equal 1; only *name*, *ipAddress*, and *settings*, fields should be returned

- *GET* `https://<Wyde bridge domain>/bridges/{ID}` – This method returns the full details about the bridge referenced by its identifier *{ID}* provided.

Parameters:

`ID` – The bridge identifier

`includeFields` – comma separated list fields that should be included in the response (e.g. *name,ipAddress*).

`excludeFields` – comma separated list fields that should be excluded from the response (e.g. *settings*).

Returns Data:

Bridge object

Returns Status:

200 OK – returns single Bridge object according to supplied identifier

404 Not Found – object matched to the criteria is not found, i.e. the bridge with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/bridges/1"	Return the details for the bridge with *identifier 1*

- *PUT* `https://<Wyde bridge domain>/bridges/{ID}` – This method updates properties and settings of the existing bridge referenced by the identifier *{ID}* with the new information posted in the data input parameter.

Parameters:

`ID` – The identifier of the Bridge which settings should be updated

`data` – The Bridge object properties: id (can be omitted, but if specified must match to *{ID}* from the PUT URL), details, email, firstName, lastName, login (*), parented (*), password (*), phoneNumber, role (*), meetingRooms (if you would like to update bridge's meeting rooms for the bridge)

Returns Data:

The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/bridges/1*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the Bridge object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the ID from the POST URL and ID from data parameter do not match, wrong (nonexistent) ID was specified, etc.; short error description would be supplied in the plain text format

404 Not Found – object matched to the criteria is not found, i.e. the bridge with specified identifier is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X PUT --data "{\"name\":\"WYDE32\"}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/bridges/1	Update *name* of the bridge with *identifier 1* Note: for this sample we do not specify object *id* parameter in posted data
curl -v -u admin:admin -X PUT --data "{\"id\":\"1\",\"settings\":[{\"name\":\"mf_maxcalls\",\"value\":\"20\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/bridges/1	Update the *mf_maxcalls* bridge settings *value* to 20 for the bridge with *identifier 1* Note: for this sample we specify object *sid* parameter in posted data
curl -v -u admin:admin -X PUT --data "{\"settings\":[{\"name\":\"mf_maxcalls\",\"value\":\"10\"}]}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/bridges/1	Update the *mf_maxcalls* bridge settings *value* to 10 for the bridge with *identifier 1* Note: for this sample we do not specify object *id* parameter in posted data

- `GET https://<Wyde bridge domain>/settings` – This method returns the list of bridge settings which match the *filter* provided according to the specified *order*; there are two parameters *offset* and *limit* to help to implement paging on the web application.

  Parameters:

  `offset` – long zero based offset in recordset.

  `limit` – long maximum number of objects to return.

  `filter` – string criteria to use to filter the rows. The criteria should be a simple conditional statement started with one or more settings field names (settings fields *name* and *value* are permitted). Empty string or null or no *filter* parameter specified means no filter. Acceptable filter operators are listed in Table 22.

`order` – string specifying any settings field name and sort direction. Empty string or null or no *order* parameter specified means no order. The default direction is ascending and *asc* can be omitted, for descending order use *desc*.

Returns Data:

list of Attribute objects representing bridge settings

Returns Status:

200 OK – returns list of Attribute objects according to supplied range, filter and order; the returned objects represent bridge settings as the pairs of their *name* and *value*

400 Bad Request – in case of errors, e.g. the field specified in filter or order is incorrect, etc.; short error description would be supplied in the plain text format

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/settings?order=name"	Return all bridge settings ordered by their *name* ascending
curl -v -u admin:admin "http://192.168.1.32/wyderef/settings?filter=name%20eq%20agiserver_pool_size"	Return `agiserver_pool_size` bridge settings
curl -v -u admin:admin "http://192.168.1.32/wyderef/settings?filter=name%20ilike%20'billing%25'&order=name"	Return all billing settings, i.e. the settings which name is started with `billing`; sort the settings by their *name* ascending

- *GET* ***https://<Wyde bridge domain>/settings/{name}*** – This method returns the details about the specific settings referenced by the name provided.

Parameters:

name – The settings name

Returns Data:

Attribute object representing single bridge settings

Returns Status:

200 OK – returns single Attribute object according to the supplied name in form of the settings *name* and the settings *value* pair

404 Not Found – object matched to the criteria is not found, i.e. the settings with specified name is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	Description
curl -v -u admin:admin "http://192.168.1.32/wyderef/settings/agiserver_pool_size"	Return the details for the `agiserver_pool_size` bridge settings

- *PUT* ***https://<Wyde bridge domain>/settings/{name}*** – This method updates specific bridge settings with the new value posted in the data input parameter. Please note that only administrator has a permission to update the settings.

Parameters:

name – The name of the settings that should be updated

data – The settings object properties: name (can be omitted, if specified must match to the name from the POST URL), value (*)

Returns Data:

The URL (location header) to the updated object, e.g. *https://<Wyde bridge domain>/settings/agiserver_pool_size*; it is also being returned in the *Location* identifier of the response headers

Returns Status:

200 OK – the object was updated; returns the URL (location header) to the updated object

400 Bad Request – the settings object was not updated, for example if wrong *data* were specified, e.g. wrong field name or field value were specified, mandatory parameter is missing, the name from the POST URL and the name from data parameter do not match, wrong (nonexistent) name was specified, etc.; short error description would be supplied in the plain text format

404 Not Found – object matched to the criteria is not found, i.e. the settings with specified name is not found

500 Internal server error – in case of any generic server error, when an unexpected condition was encountered and no more specific message is suitable

Samples:

Call	**Description**
curl -v -u admin:admin -X PUT --data "{\"value\":10}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/settings/agiserver_pool_size	Update `agiserver_pool_size` bridge settings *value* to 10 (omit the settings *name* in *data* parameter)
curl -v -u admin:admin -X PUT --data "{\"name\":\"agiserver_pool_size\",\"value\":5}" --header "Content-Type:application/json" http://192.168.1.32/wyderef/settings/agiserver_pool_size	Update `agiserver_pool_size` bridge settings *value* to 5 (specify the settings *name* `agiserver_pool_size` in *data* parameter)