API Documentation

Available API Methods
Sessions
GET session
GET sessions
PUT session
POST session
DELETE session
GET invitees
PUT invitee
POST invitee
DELETE invitee
Recordings
GET recording
GET recordings
POST recording
DELETE recording
PUT file (URL's only at the moment)

Quickstart
The Onstream API is a RESTful Web Service.
It accepts requests using standard HTTP methods, such as GET, PUT, POST and DELETE. Currently, "Basic HTTP Authentication" is the only available method of authenticating. We are working to additionally provide "Digest HTTP Authentication".
Examples given here are provided for the PHP programming language and uses cURL to make the HTTP requests.
Important points:
1. According to the HTTP specification, when making GET requests, all parameters must be passed as URL/querystring parameters.
2. When making PUT, POST or DELETE requests, all parameters must be passed in the request body; URL/querystring parameters in such requests will be ignored, except for the format variable.
3. The data format of the request body must either:
a. contain name/value pairs such as id=1&var2=val2 or,
b. contain an input_type and rest_data variable, the former must be set to "json" and the latter must be a valid JSON array, such as
input_type=json&rest_data={"id":"1","var2":"val2"}
Most developers will find the JSON approach more robust as it allows the passing of objects and arrays whilst also being simpler to implement (simply json_encode() your data array before sending it). We provide PHP code examples for how requests are made with each API method below.

We are working to allow for additional input_type (such as XML), but for the time being you must always set this to “json”.
There is more choice available for return formats - take a look at the section called “Return Formatting” below.

How to use HTTP Basic Auth while developing and debugging
To access the Onstream API methods you must pass the Authorization header as part of each request, set it to basic and pass the required encrypted string. For example:
Authorization: Basic bWF0dGhpYXM6YWWtaW4=
The encrypted string must be a base64 encoded combination of username and password in the following format:
myUsername:myPassword
In Fiddler you can create this string by opening the Encoder tab, selecting base64 and entering your credentials.
Essential Tools:
Fiddler2 - HTTP Web Debugger. Allows you to send GET, PUT, POST, DELETE and other HTTP requests, modify headers and request body, and capture the return values. Essential for testing and debugging your API integration.
Essential Reading:
“RESTful Web Services” by Leonard Richardson & Sam Ruby (O’Reilly) - some formulations in this document have been borrowed from this excellent book, in particular for the section on HTTP Error Codes.

General Definitions
[ver] - The version of the API endpoint, must be defined in each request URL. The current version is 2.
Example: GET http://domain.com/api/2/[username]/authverify
[username] - The username of the account which is making the API requests.
Example: GET http://domain.com/api/[ver]/johndoe/authverify

HTTP Error Codes
400 “Bad Request”
Returned when providing input during a PUT operation which would leave the resource in an incomplete or inconsistent state. The provided input is nonsensical or corrupt.
Example: Providing an alphabetic value of “abcd” for an integer parameter, such as “offset”.
401 “Unauthorized”
Returned when trying to operate on a protected resource without providing the proper authentication credentials (either wrong or none at all). The response header WWW-Authenticate describes what kind of authentication the server will accept.
404 “Not Found”
Returned when trying to access a URI that does not correspond to any existing resource. The only possible exception is when a client is trying to PUT a new resource to that URI, see 201.
Example: Trying to retrieve a non-existing user.
405 “Method Not Allowed”
Returned when the client tries to use an HTTP method which this particular resource doesn’t support.
Example: Trying to PUT or DELETE a read-only resource.

409 "Conflict"
Returned when providing input during a PUT operation which would cause the resource state to conflict with some other resource.
Example: Trying to change your username to a name that's already taken. Trying to delete a folder that is not empty.
410 "Gone"
Returned when the server knows there used to be a resource there, but there isn't anymore. Also see 404, when the server has no idea about the resource at all.
500 "Internal Server Error"
There is a problem on the server side.

HTTP Success Codes
200 "OK"
Returned when a GET operation is successful, a POST operation to append to an existing resource is successful, and when a DELETE operation is successful.
201 "Created"
When a successful PUT or POST (when creating new) operation is carried out a 201 is returned, along with the new URI in the Location header. See 400 and 409 for related error codes.
301 "Moved Permanently"
Sometimes a PUT operation will change the URI of the resource just modified. In such cases, a 301 is returned, along with the new URI in the Location header. Future requests to the old URI will result in a 301, 404 or 410 return.

Return Formatting
The Onstream API supports a number of different return formats.
By appending /format/[value] to the URLs of the API method calls, or by passing the correct MIME-type, you can choose the format of the returned data. These may differ for each call, so, for example you may wish to obtain a list of users in XML format, but get a list of invitees returned in JSON.
Available formats and how to request them:

Format	URL Parameter	Sample return
xml	none (default) or /format/xml	<?xml version="1.0" encoding="utf-8"?> <xml> <item> <id>1</id> <login>username</login> <email>email@domain.com</email> <first_name>First Name</first_name> <last_name>Last Name</last_name> [...] </item> </xml>
json	/format/json	[{ "id": "1",

		"login": "username", "email": "email@domain.com", "first_name": "First Name", "last_name": "Last Name", [...] }]

serialize	/format/serialize	a:1:
		{
		i:0;a:5:
		{
		s:2:"id";
		s:1:"1";
		s:5:"login";
		s:8:"username";
		s:5:"email";
		s:16:"email@domain.com";
		s:10:"first_name";
		s:10:"First Name";
		s:9:"last_name";
		s:9:"Last Name";
		[...]
		}
		}

php (can be used in eval)	/format/php	array (0 => array ('id' => '1', 'login' => 'username', 'email' => 'email@domain.com', 'first_name' => 'First Name', 'last_name' => 'Last Name', [...]))
html	/format/html	<table border="0" cellpadding="4" cellspacing="0"> <tr> <th>id</th> <th>login</th> <th>email</th> <th>first_name</th> <th>last_name</th> <th>[...]</th> </tr> <tr> <td>1</td>

		<td>username</td> <td>email@domain.com</td> <td>First Name</td> <td>Last Name</td> <td>[...]</td> </tr> </table>
csv	/html/csv	id,login,email,first_name,last_name,[...] 1,"username","email@domain.com","First Name","Last Name",[...]

Method	GET authverify
URL	/api/[ver]/[username]/authverify
Description	Use to test if the auth credentials passed with HTTP Basic are valid

Arguments		
Required	none	
Optional	none	
HTTP	GET	
Return	authenticated (bool)	If 1/true, credentials are valid.

Example	
Request	GET http://domain.com/api/[ver]/[username]/authverify
Request Body	none
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <authenticated>1</authenticated> <message>You are authorized to use the API</message> </xml>

PHP	<?php
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array();
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/authverify');
	curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'PUT');
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);

```
$result = curl_exec($curl);
curl_close($curl);
echo $result;
?>
```

Method	GET session(s)
URL	/api/[ver]/[username]/session/id/[session_id] /api/[ver]/[username]/sessions
Description	Get one or many sessions associated with this user account. /session will return a single session dataset. Requires id. /sessions returns multiple results.
Arguments	
Required	id for /session none for /sessions

Optional	return(str)	Choose which fields should be returned by adding a list of semi-colon delimited field names eg .../sessions/return/id;topic;start_time If not defined, a pre-defined subset of fields is returned (see sample XML below)

	Available only when retrieving multiple results with /sessions

	count(int)	Specify number of returned results. Default is 100.
	offset(int)	Specify to move the cursor through the recordset. Default is 0.
	order(str)	Return results in ascending or descending order of session ID. Default is asc. Accepted values: asc, desc

	Available only when retrieving a single result with /session

	id(int)	id of the session to be returned. If defined then count, offset and order are ignored.
HTTP	GET	

Return	id (int)	The id of the session.
	topic (str)	The name or title of this session.
	start_time (datetime)	The datetime string of when this session will take or has taken place. Will always default to the user's timezone & DST settings.See the "Timezones" section for more info.
	duration (int)	Duration of the meeting, in minutes.
	password (str)	The session's access password, required to

		join. None, if empty.
	session_access_code (int)	Code to join the session via telephone. Only available if VOIP bridge module is purchased.
	session_link (str)	Full link to the session. If friendly_url is NULL, session_key is used.
	custom_data(arr)	An array of objects containing custom fields in the form of custom1, custom2 etc

Example	
Request	GET http://domain.com/api/[ver]/[username]/session/id/1
Request Body	none
Response	<?xml version="1.0" encoding="UTF-8"?> <root> <sessions> <session> <id>1</id> <topic>My first session</topic> <start_time>2010-08-31 14:00:00</start_time> <duration>60</duration> <timezone>UP2</timezone> <password>some-password</password> <session_link>http://domain.com/user/friendly_url-or-session_key</session_link> <custom_data> <custom1>first custom value</custom1> <custom2>second custom value</custom2> <custom3>third custom value</custom3> <custom4>...</custom4> </custom_data> </session> [...] </sessions> </root>

PHP	<?php
	// GET session
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/session/id/1');
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

Method	PUT session
URL	/api/[ver]/[username]/session
Description	Create a new session.

Arguments		
Required	topic (str)	The title of the meeting.
	duration (int)	Duration of the meeting, in minutes.
	start_time (datetime)	Start time in datetime format eg "2010-08-31 11:05:00". Will always default to the user's timezone & DST settings.See the "Timezones" section for more info.
Optional	friendly_url (str)	Define a friendly URL segment to be used instead of auto-generated rather than encrypted meeting ID.
	password (str)	To password protect the meeting define this.

	invited_participants (arr)	JSON array of users who should receive an email invitation, in the following format:
		[
		{
		"id": "1",
		"email": "email@domain.com",
		"first_name": "First Name",
		"last_name": "Last Name",
		"role": 1,
		"send_email_invitation": false
		}
		]
		role (int): Defines the user's status in the meeting: Moderator (1), Participant (2) or Observer (3)
		
		send_email_invitations (bool): Set to false (or 0) to avoid sending a server-generated email invitation to the user. Default is true. Email invitations will be sent as soon as this call is made. If you want to send your own invitation emails, use GET invitees to obtain the personal_session_link for each user.

	custom_data(arr)	JSON array of custom data, in the following formats: [{ "custom1": "custom value 1", "custom2": "custom value 2", "custom3": "custom value 3"

		}] You may add additional custom fields (custom4 and beyond), provided the field name is in the format “customX” where X is an integer incremented by 1. Custom value is varchar(200).
HTTP	PUT	
Return	id (int)	The id of the newly created session.

Example	
Request	PUT http://domain.com/api/[ver]/[username]/session
Request Body	input_type=json&rest_data={"topic":"Test","duration":60, "start_time":"2010-09-10 12:00:00","timezone":"UP1", "invited_participants":[{"email":"email@domain.com","first_name":"First Name","last_name":"Last Name","role":1,"send_email_invitation":0}],"custom_data": [{"custom1":"first custom value", "custom2":"second custom value", "custom3":"third custom value","custom4":"fourth custom value"}]}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Session added</message> </xml>
PHP	<?php // PUT session $username = 'username'; $password = 'password'; $baseurl = 'http://domain.com/api/1/'; $parameters = array('topic' => 'Test', 'duration' => 60, 'start_time' => '2010-10-09 12:00:00', 'timezone' => 'UP1', 'invited_participants' => array(array('email'=>'email@domain.com', 'first_name'=>'First Name', 'last_name'=> 'Last Name', 'role'=> 1), array('email'=>'email2@domain.com', 'first_name'=>'First Name2', 'last_name'=> 'Last Name2', 'role'=> 2)));

```
// encode as JSON
$json = json_encode($parameters);
$postArgs = 'input_type=json&rest_data=' . $json;
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL,  $baseurl.$username.'/session');
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'PUT');
curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
$result = curl_exec($curl);
curl_close($curl);
echo $result;
?>
```

Method	POST session
URL	/api/[ver]/[username]/session
Description	Edit an existing session.

Arguments		
Required	id (int)	The id of the session.
Optional	topic (str)	The title of the meeting.
	duration (int)	Duration of the meeting, in minutes.
	start_time (datetime)	Start time in datetime format eg “2010-08-31 11:05:00”. Will always default to the user’s timezone & DST settings.See the “Timezones” section for more info.
	friendly_url (str)	Define a friendly URL segment to be used instead of auto-generated rather than encrypted meeting ID.
	password (str)	To password protect the meeting define this.

	invited_participants (arr)	JSON array of users who should receive an email invitation, in the following format:
		[
		{
		"id": "1",
		"email": "email@domain.com",
		"first_name": "First Name",
		"last_name": "Last Name",
		"role": 1,
		"send_email_invitation": false
		}
		]
		role (int): Defines the user’s status in the meeting: Moderator (1), Participant (2) or Observer (3)

		send_email_invitations (bool): Set to false (or 0) to avoid sending a server-generated email invitation to the user. Default is true. Email invitations will be sent as soon as this call is made. If you want to send your own invitation emails, use GET invitees to obtain the personal_session_link for each user.

	custom_data(arr)	JSON array of custom data, in the following formats:
		[
		{
		"custom1": "custom value 1",
		"custom2": "custom value 2",
		"custom3": "custom value 3"
		}
		]
		You may add additional custom fields (custom4 and beyond), provided the field name is in the format "customX" where X is an integer incremented by 1.
		Custom value is varchar(200).

HTTP	POST	
Return	id (int)	The id of the edited session.
	message (str)	"Session updated".

Example	
Request	POST http://domain.com/api/[ver]/[username]/session
Request Body	input_type=json&rest_data={"id":1,"topic":"Test","duration":60, "start_time":"2010-09-10 12:00:00","timezone":"UP1", "invited_participants":[{"email":"email@domain.com","first_name":"First Name","last_name":"Last Name","role":1,"send_email_invitation":0}],"custom_data": [{"custom1":"first custom value", "custom2":"second custom value", "custom3":"third custom value","custom4":"fourth custom value"}]}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Session updated</message> </xml>
PHP	<?php // POST session $username = 'username'; $password = 'password'; $baseurl = 'http://domain.com/api/1/'; $parameters = array('id' => 1, 'topic' => 'Test',

```
    'duration' => 60,
    'start_time' => '2010-09-10 12:00:00',
    'timezone' => 'UP1',
    'invited_participants' => array(
     array(
            'email'=>'email@domain.com',
            'first_name'=>'First Name',
            'last_name'=> 'Last Name',
            'role'=> 1,
            'send_email_invitation'=> 0
        ),
     array(
            'email'=>'email2@domain.com',
            'first_name'=>'First Name2',
            'last_name'=> 'Last Name2',
            'role'=> 2
        )
      )
    );
// encode as JSON
$json = json_encode($parameters);
$postArgs = 'input_type=json&rest_data=' . $json;
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL,  $baseurl.$username.'/session');
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
$result = curl_exec($curl);
curl_close($curl);
echo $result;
?>
```

Method	DELETE session
URL	/api/[ver]/[username]/session
Description	Delete an existing session.

Arguments		
Required	id (int)	The id of the session to be deleted. Note: the id may also be appended to the URL (.../id/xxx) instead of in the request body.
Optional	none	
HTTP	DELETE	
Return	id (int)	The id of the deleted session.
	message (str)	“Session deleted”.

Example	
Request	DELETE http://domain.com/api/[ver]/[username]/session
Request Body	id=1 or input_type=json&rest_data={"id":"1"}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Session was deleted</message> </xml>

PHP	<?php
	// DELETE session
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array(
	'id' => 1
	);
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/session');
	curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'DELETE');
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

Method	GET invitees
URL	/api/[ver]/[username]/invitees/id/1/role/moderators
Description	Get data of people invited to a session. All invitees are grouped into either moderators, participants or observers, depending on the role given to them when creating the session.

Arguments		
Required	id (int)	The id of the session for which invitees should be retrieved.
Optional	role (string)	Limit the returned invitees to users holding a particular role in the session. Possible values: all, moderators, participants, observers.

HTTP	GET	
Return	id (int)	Account id of the invitee.
	first_name (str)	The first name of the invited user.
	last_name (str)	The last name of the invited user.
	email (str)	The email address of the invited user.
	personal_session_link (string)	A custom session link for this particular user. It is different from the session_link in GET sessions in that it does not prompt for a login, and will put the user into the session (with his username) with a single click.

Example	
Request	GET http://domain.com/api/[ver]/[username]/invitees/id/1/role/moderators
Request Body	none
Response	<?xml version="1.0" encoding="UTF-8"?> <xml> <moderators> <item> <id>4</id> <first_name>Cesc</first_name> <last_name>Fabregas</last_name> <email>cesc@arsenal.com</email> <personal_session_link>http://.../[user]/go/ABC123</personal_session_link> </item> <item> [...] </item> </moderators> <participants> [...] </participants> <observers> [...] </observers> </xml>

PHP	<?php
	// GET invitees
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL,
	$baseurl.$username.'/invitees/id/1/role/moderators');
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);

```
$result = curl_exec($curl);
curl_close($curl);
echo $result;
?>
```

Method	PUT invitee
URL	`/api/[ver]/[username]/invitee`
Description	Add an invitee to a session.

Arguments		
Required	session_id (int)	ID of the session to which invitee is added.
	email (str)	Email address of the invitee.
Optional	first_name (str)	First name of the invitee.
	last_name (str)	Last name of the invitee.
	send_email_invitation (bool)	Set to false (or 0) to avoid sending a server-generated email invitation to the user. Default is true. Email invitations will be sent as soon as this call is made. If you want to send your own invitation emails, use GET invitees to obtain the personal_session_link for each user.
	role(int)	Define role of the invitee in the session. Default is 2. 1 = Moderator, 2 = Participants, 3 = Observer
HTTP	PUT	
Return	id (int)	The user_id of the newly created invitee.

Example	
Request	PUT http://domain.com/api/[ver]/[username]/invitee
Request Body	input_type=json&rest_data={"session_id":1,"email":"robin@arsenal.com","first_name":"Robin","last_name":"van Persie","send_email_invitation":false,"role":1}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Invitee added</message> </xml>
PHP	<?php // PUT invitee $username = 'username'; $password = 'password'; $baseurl = 'http://domain.com/api/1/'; $parameters = array('session_id' => 1, 'email' => 'robin@arsenal.com',

```
    'first_name' => 'Robin',
    'last_name' => 'van Persie',
    'send_email_invitation' => 0,
    'role' => 1
    );
// encode as JSON
$json = json_encode($parameters);
$postArgs = 'input_type=json&rest_data=' . $json;
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL,  $baseurl.$username.'/invitee');
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'PUT');
curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
$result = curl_exec($curl);
curl_close($curl);
echo $result;
?>
```

Method	POST invitee
URL	`/api/[ver]/[username]/invitee`
Description	Modify the role of an invitee to a session.

Arguments		
Required	session_id (int)	ID of the session to which invitee belongs.
	user_id(int)	User ID of the invitee. The PUT call returned this.
Optional	send_email_invitation (bool)	Set to false (or 0) to avoid sending a server-generated email invitation to the user. Default is true. Email invitations will be sent as soon as this call is made. If you want to send your own invitation emails, use GET invitees to obtain the personal_session_link for each user.
	role(int)	Define role of the invitee in the session. Default is 2. 1 = Moderator, 2 = Participants, 3 = Observer
HTTP	POST	
Return	id (int)	The user_id of the modified invitee.

Example	
Request	POST http://domain.com/api/[ver]/[username]/invitee
Request Body	input_type=json&rest_data={"session_id":1,"user_id":1,"send_email_invitation":false,"role":1}

Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Invitation was updated</message> </xml>

PHP	<?php
	// POST invitee
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array(
	'session_id' => 1,
	'user_id' => 1,
	'send_email_invitation' => 0,
	'role' => 1
	);
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/invitee');
	curl_setopt($curl, CURLOPT_POST, true);
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

Note: To edit email, first_name and last_name of invitees, use the POST user method.

Method	DELETE invitee
URL	`/api/[ver]/[username]/invitee`
Description	Delete an invitee from a session.

Arguments		
Required	session_id (int)	ID of the session to which invitee is added.
	user_id (int)	User ID of the invitee.
HTTP	DELETE	
Return	nothing	

Example	
Request	DELETE http://domain.com/api/[ver]/[username]/invitee
Request	input_type=json&rest_data={"session_id":1,"user_id":1}

Body	
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Invitee deleted</message> </xml>

PHP	<?php
	// DELETE invitee
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array(
	'session_id' => 1,
	'user_id' => 1
	);
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/invitee');
	curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'DELETE');
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

Method	GET recording(s)
URL	/api/[ver]/[username]/recording/id/[recording_id] /api/[ver]/[username]/recordings
Description	Get one or many recordings associated with this user account. /recording will return a single recording dataset. Requires id. /recordings returns multiple results.
Arguments	
Required	id for /recording none for /recordings

Optional	return(str)	Choose which fields should be returned by adding a list of semi-colon delimited field names eg .../recordings/return/id;title;duration If not defined, a pre-defined subset of fields is returned (see sample XML below)

	Available only when retrieving multiple results with /recordings

	session(int)	pass a session id to limit results to recordings made in that session only. Otherwise recordings made across all sessions are returned.
	count(int)	Specify number of returned results. Default is 100.
	offset(int)	Specify to move the cursor through the recordset. Default is 0.
	order(str)	Return results in ascending or descending order of recording ID. Default is asc. Accepted values: asc, desc

	Available only when retrieving a single result with /recording

	id(int)	id of the recording to be returned. If defined then count, offset and order are ignored.
HTTP	GET	

Return	id (int)	The id of the recording.
	session_id (int)	The id of the session in which this recording was made.
	title (str)	The name or title of this recording.
	description (str)	Description of the recording.
	password (str)	Password required to view the recording. Inherited from it's session, if set there, none, if empty.
	duration (int)	Length of recording, in milliseconds.
	allow_anonymous_playback (bool)	If true, users will not be prompted to provide a screenname when viewing a recording. If a password is set, they will be prompted regardless of this value.
	recording_link (str)	Full link to the recording.
	creation_date (datetime)	Datetime stamp when the recording was made.

Example	
Request	GET http://domain.com/api/[ver]/[username]/recording/id/1
Request Body	none
Response	<?xml version="1.0" encoding="UTF-8"?> <root> <recordings> <recording> <id>1</id> <session_id>1</session_id>

	<title>My first recording</title> <description>This is my first recording</description> <password>some-password</password> <duration>25880</duration> <allow_anonymous_playback>1</allow_anonymous_playback> <recording_link>http://domain.com/play/1-My-first-recording</recording_link> <creation_date>2010-08-31 14:00:00</creation_date> </recording> [...] </recordings> </root>

PHP	<?php
	// GET recording
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL,
	$baseurl.$username.'/recording/id/1');
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

NOTE: PUT recording is not available through the API. Recordings are always made from within Onstream only.

Method	POST recording
URL	/api/[ver]/[username]/recording
Description	Edit an existing recording.

Arguments		
Required	id (int)	The id of the recording.
Optional	title (str)	The title of the recording.
	description (str)	The description of the recording.
	password (str)	Set a password to view this recording. Leave empty, for open access.
	allow_anonymous_playback (bool)	Set to true (or 1) to supress login screen to the recording. Ignored if password is set.
HTTP	POST	
Return	id (int)	The id of the edited recording.
	message (str)	“Recording updated”.

Example	

Request	POST http://domain.com/api/[ver]/[username]/recording
Request Body	input_type=json&rest_data={"id":1,"title":"My recording","description":"More info about my recording", "password":"some-password"}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Recording updated</message> </xml>

PHP	<?php
	// POST recording
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array(
	'id' => 1,
	'title' => 'My recording',
	'description' => 'More info about my recording',
	'password' => 'some-password',
	'allow_anonymous_playback' => 1,
	);
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/recording');
	curl_setopt($curl, CURLOPT_POST, true);
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

Method	DELETE recording
URL	/api/[ver]/[username]/recording
Description	Delete an existing recording.

Arguments		
Required	id (int)	The id of the recording to be deleted. Note: the id may also be appended to the URL (.../id/xxx) instead of in the request body.
Optional	none	
HTTP	DELETE	
Return	id (int)	The id of the deleted recording.
	message (str)	“Recording deleted”.

Example	
Request	DELETE http://domain.com/api/[ver]/[username]/recording
Request Body	id=1 or input_type=json&rest_data={"id":"1"}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>Recording was deleted</message> </xml>

PHP	<?php
	// DELETE session
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array(
	'id' => 1
	);
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/recording');
	curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'DELETE');
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

EXPERIMENTAL SECTION:
Note: All calls below this point are in development. Do not yet use in production.
PUT file: Currently you can only add URL's, not physical files. Also, all added URL's are stuck into the user's Media Library base directory, until the GET directory call is completed.

Method	PUT file
URL	/api/[ver]/[username]/file
Description	Create a new file in the Media Library.

Arguments		
Required	file_link (str)	The full URL
	is_url (int)	Whether the file is a URL. By default this is

		0
Optional	dir_id (int)	The ID of the directory which to create the file in. **Currently not implemented**
	file_description (str)	Description of the file, max 255 chars
HTTP	PUT	
Return	id (int)	The id of the newly created file.
	message (str)	“File added”.

Example	
Request	PUT http://domain.com/api/[ver]/[username]/file
Request Body	input_type=json&rest_data={"file_link":"http://www.domain.com/myFile.swf?someParam=123","is_ is my flash movie."}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>File added</message> </xml>

PHP	<?php
	// PUT user
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array(
	'file_link' => 'http://www.domain.com/myFile.swf?someParam=123',
	'is_url' => '1',
	'file_description'=>'This is my flash movie.'
	);
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/file');
	curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'PUT');
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;
	?>

Note: This method call is in development. After calling PUT file, call PUT file-in-session if you want to associate the file with a session (required when auto-opening a file). Currently also only works with URL’s.

Method	PUT file_in_session

URL	/api/[ver]/[username]/file_in_session
Description	Associate an existing file or URL with a session, by adding it to the FileShare component.

Arguments		
Required	file_id (str)	The ID of an existing file
	session_id (int)	The ID of an existing session
Optional	is_auto_opening (int)	Whether the file should load automatically when user enters the session. By default this is NULL.
HTTP	PUT	
Return	id (int)	The id of the newly created File-Share association.
	message (str)	“File added to session”.

Example	
Request	PUT http://domain.com/api/[ver]/[username]/file_in_session
Request Body	input_type=json&rest_data={"file_id":"1","session_id":"1","is_auto_opening":"1"}
Response	<?xml version="1.0" encoding="utf-8"?> <xml> <id>1</id> <message>File added to session</message> </xml>

PHP	<?php
	// PUT user
	$username = 'username';
	$password = 'password';
	$baseurl = 'http://domain.com/api/1/';
	$parameters = array(
	'file_id' => '1',
	'session_id' => '1',
	'is_auto_opening' => '1'
	);
	// encode as JSON
	$json = json_encode($parameters);
	$postArgs = 'input_type=json&rest_data=' . $json;
	$curl = curl_init();
	curl_setopt($curl, CURLOPT_URL, $baseurl.$username.'/file_in_session');
	curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'PUT');
	curl_setopt($curl, CURLOPT_POSTFIELDS, $postArgs);
	curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
	curl_setopt($curl, CURLOPT_USERPWD, $username.':'.$password);
	$result = curl_exec($curl);
	curl_close($curl);
	echo $result;

	?>

Timezones

How it works

All session start times are based on the session owner's defined timezone and DST setting, so make sure to set this correctly with PUT/POST user.
Onstream stores all session start times as UTC+0 as a baseline, and will calculate the session start time for each user that is involved in this session (invited guest, invited registered users, the session owner himself) according to that user's timezone and DST settings.
For example, if you set your account timezone to UTC+1 Berlin and enable DST the following will happen when you schedule a meeting for 18:00 -
1. Onstream will store the meeting to start at 16:00 UTC+0
2. Any invited registered user will see the meeting in his timezone and DST setting so a user UTC-5 in EST with DST enabled will see the meeting scheduled for 12:00 - in his account center, in his email invitation and also when retrieving this session via the API GET session method.
3. Any invited guest user (who does not have a timezone or DST setting) will receive an email invitation with the timezone and DST settings defaulted to the session owner.
API parameters
Use the following timezone parameters. The numbers refer to hours offset from UTC, the letter M in UM refers to "minus" and the letter P in UP refers to "plus". Therefore UM5 is the same as UTC-5, and UP1 is the same as UTC+1.
UM12 for International Date Line West.
UM11 for Midway Island, Samoa.
UM10 for Hawaii-Aleutian Standard Time (US & Canada).
UM95 for Taiohae, Marquesas Islands.
UM9 for Alaskan Standard Time (US & Canada).
UM8 for Pacific Standard Time (US & Canada).
UM7 for Mountain Standard Time (US & Canada).
UM6 for Central Standard Time (US & Canada).
UM5 for Eastern Standard Time (US & Canada).
UM45 for Venezuelan Standard Time.
UM4 for Atlantic Standard Time (Canada).
UM35 for St. John`s, Newfoundland and Labrador.
UM3 for Brazil, Buenos Aires, Georgetown.
UM2 for Mid-Atlantic.
UM1 for Azores, Cape Verde Islands.
UTC for Western Europe Time, London, Lisbon, Casablanca.
UP1 for Berlin, Brussels, Copenhagen, Madrid, Paris.
UP2 for South Africa.
UP3 for Baghdad, Riyadh, Kaliningrad.
UP35 for Tehran.
UP4 for Moscow, St. Petersburg, Abu Dhabi, Muscat, Baku, Tbilisi.
UP45 for Kabul.
UP5 for Islamabad, Karachi, Tashkent.
UP55 for Bombay, Calcutta, Madras, New Delhi.
UP575 for Kathmandu.
UP6 for Ekaterinburg, Almaty, Dhaka, Colombo.
UP65 for Yagoon.
UP7 for Omsk, Bangkok, Hanoi, Jakarta.
UP8 for Krasnoyarsk, Beijing, Perth, Singapore, Hong Kong.
UP875 for Western Australia, Caiguna, Eucla.
UP9 for Irkutsk, Tokyo, Seoul, Osaka, Sapporo.
UP95 for Adelaide, Darwin.

UP10 for Yakutsk, Eastern Australia, Guam.
UP105 for Lord Howe Island (Australia).
UP11 for Vladivostok, Solomon Islands, New Caledonia.
UP115 for Norfolk Island.
UP12 for Magadan, Auckland, Wellington, Fiji.
UP1275 for Chatham Island.
UP13 for Tonga.
UP14 for Kiribati.