Server to Server Web API Guide

November 6th, 2018

Introduction

This document will describe how to use the Vivox Server to Server Web API to control the players' experience in a few specific ways, as well as enable game clients to securely access the Vivox system.

Prerequisites

The following are prerequisites to using the Vivox Server to Server Web API:

- Your assigned Vivox API End-Point.
- Your assigned Admin username
- Your assigned Admin password.

The API end-point, admin username, and admin password is provided when you create an application on the Vivox Developer Portal.

Security and Authentication

The Vivox Server to Server Web API has the following requirements:

1. All access to the Server to Server Web API must be by an authenticated user.
2. All access to the Server to Server Web API must use the HTTPS protocol.

The Vivox Server to Server Web API will determine if a request is being made by an authenticated user by the authentication token passed as the *auth_token* parameter to the request. The *auth_token* value is returned from the *viv_signin.php* call.

Access Tokens

For more information on Vivox Access Tokens, see the Access Token Developer Guide found in the Epic Documentation.

API Documentation

Common Parameters

Several URL parameters are common to all or nearly all of the APIs.

Parameter	Meaning	Comments

userid	The Vivox login username of the requester of the Vivox operation.	This is found in your VIVOX API INFO on the Developer Portal.
pwd	The password of the user identified by the 'userid' parameter.	This is found in your VIVOX API INFO on the Developer Portal.
auth_token	An explicit authorization token (cookie).	This value is contained in the return XML from the viv_signin call.
user_uri	The SIP URI of the user that is the target of the operation.	This value must be URL encoded when calling the API.
channel_uri	The SIP URI of the channel that is the target of the operation.	This value must be URL encoded when calling the API.
access_token	The access token with proper claims for this action	Details on implementing Access Token support can be found in the Access Token Developer Guide.

Login */api2/viv_signin.php*

/api2/viv_signin.php is used to sign in to the Vivox Service and obtain an *auth_token*. The *auth_token* will be used to authenticate subsequence web requests.

Required Parameters

- *userid* - The Vivox login name of the administrative user to be logged in
- *pwd* - The password associated with the userid.

Return Codes

- 200 - Wrong user credentials.
- 300 - Required parameter(s) missing or invalid.
- 1600 - Login Failed. Unknown Account.
- 1602 - Login Failed. Account has not been activated.
- 1603 - Login Failed. Account has been disabled.
- 1604 - Login Failed. Please contact support.

Notes

- Returns an *auth_token* in the response on success.

- The *auth_token* may be used by all other APIs in place of userid and pwd parameters.

Example

```
https://vdx5.www.vivox.com/api2/viv_signin.php?userid=api_tester&pwd=APItest
```

Response

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<response>
     <level0>
          <status>OK</status>
          <body>
               <status>OK</status>
<auth_token>api_tester:1278945699:220a34213b4273a1eab1ac18b5fe67b6:TesterAlias:1</auth_token>
          </body>
     </level0>
</response>
```

Control active channels /api2/viv_chan_cmd.php

This API is used by the game server to control various aspects of the audio experience of users in a specific channel. It does this performing the following operations:

- Mute a specific user for all other users in a specific channel
- Unmute a specific user for all other users in a specific channel
- Mute all the users in a specific channel
- Unmute all the users in a specific channel
- Drop (aka kick) all the users from a specific channel
- Drop (aka kick) a specific user from a specific channel

The application controls which operation is performed by specifying a "mode" parameters with a specific value. The following table lists the valid mode parameters values, and any additional parameters for that API call.

Operation	Mode Value	Required Parameters	Optional Parameters

Mute Specific User	mute	auth_token - the token returned from viv_signin user_uri - the user's URI chan_uri - the channel's URI access_token - the access token with proper claims for this action	scope - one of "text", "audio", or "both" with a default of "audio"
Unmute Specific User	unmute	auth_token - the token returned from viv_signin user_uri - the user's URI chan_uri - the channel's URI access_token - the access token with proper claims for this action	scope - one of "text", "audio", or "both" with a default of "audio"
Mute All Users	mute_all	auth_token - the token returned from viv_signin chan_uri - the channel's URI access_token - the access token with proper claims for this action	scope - one of "text", "audio", or "both" with a default of "audio"
Unmute All Users	unmute_all	auth_token - the token returned from viv_signin chan_uri - the channel's URI access_token - the access token with proper claims for this action	scope - one of "text", "audio", or "both" with a default of "audio"
Drop/Kick A Specific User	kick	auth_token - the token returned from viv_signin user_uri - the user's URI chan_uri - the channel's URI access_token - the access token with proper claims for this action	
Drop All Users	drop_all	auth_token - the token returned from viv_signin	

		chan_uri - the channel's URI access_token - the access token with proper claims for this action	

Error Handling

If an error is encountered while making a web call, an XML document will be returned with the status element set to “ERR” and the code and msg elements set respectively. The following is an example XML error document.

```
<response xmlns="http://www.vivox.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="/xsd/error.xsd">
     <level0>
          <status>ERR</status>
          <body>
               <code>200</code>
               <msg>Wrong user credentials.</msg>
          </body>
     </level0>
</response>
```

The following is a list of common error codes:

- 200 - Wrong User credentials.
- 201 - Expired user credentials.
- 202 - Missing user credentials.
- 212 - Account not allowed to perform operation.
- 403 - Account does not exist.
- 300 - Required parameter(s) missing or invalid.
- 301 - Unknown mode.
- 302 - Unable to parse argument.
- 807 - Channel not found.
- 718 - Failed changing conference media.
- 1600 - Login Failed. Unknown Account.
- 1602 - Login Failed. Account has not been activated.
- 1603 - Login Failed. Account has been disabled.

- 1604 - Login Failed. Please contact support.

Using Fully Qualified Identifiers

You will need to ensure that the *user_uri* and *chan_uri* parameters are fully qualified URIs. A fully qualified URI will include your issuer as part of the URI. The specific fully qualified URI format is as follows:

- Users - `sip:.issuer.userid.@somedomain.vivox.com`
- Admins - `sip:Issuer-AdminID@somedomain.vivox.com`
- Channels - `sip:confctl-[g/d/e]-issuer.channelid@somedomain.vivox.com`

Note: Admin IDs do not follow the same format as signed in users.

Extended Syntax for 3D Channel Parameters

Developers can optionally specify parameters for positional (`-d-`) channels, using an extended syntax in the channelid portion of the channel URI. Immediately following the channelid, before the "`@`" you add "`!p-`" followed hyphen-delimited parameters indicating 3D channel property values, in the following format:

```
!p-{max_range}-{clamping_distance}-{rolloff}-{distance_model}

{max_range} - integer, must be > 0
{clamping_distance} - integer, must be 0 <= clamping_distance <=
max_range
{rolloff} - float, must be >= 0.0
{distance_model} - integer, 1 = inverse-clamped, 2 = linear-clamped,
3 = exponent-clamped
```

Example:

```
sip:confctl-d-issuer.channelid!p-30-1-1.0-1@somedomain.vivox.com
```