EZserver API Guide

www.ezhometech.com

Updated :09/02/2014

Version : 1.0.1

Content

1. Introduction

EZserver API is a simple way to let Apps or Content Management System interact with EZserver. It includes a set of REST APIs to be integrated into Apps or Content Management System.

1.1 Features:

- Suitable for Javascript Applications
- Suitable for PHP Applications
- Suitable for C/C++ Applications in iOS devices
- Suitable for Java Applications in Android devices

1.2 Who will use API?

- Web developers
- iOS developers
- Android Developers
- IPTV CMS developers

1.3 What should I have?

- Linux or Windows
- EZserver free or prof. version
- Your applications
- IDE tools

1.4 API Examples

- EZwplayer
- EZflvplayer
- EZsplayer

2. Security Token APIs

Security Token APIs have *createtoken, createtokenbased64* and *destroytoken* APIs. An application needs to get a token from EZserver first, then it uses the token to do the following sequential HTTP APIs. This initial HTTP API is *createtoken* with user id and password or *createtokenbased64* with base64-encoded string of user id and password. And the final HTTP API is *destroytoken* with the token, user id and password.

- **createtokenbased64**

Description: The initial HTTP API is to get the authorization token.

API syntax: GET HTTP/1.1 /token/createtokenbased64?encrpty=xxx

Parameter:

- encrpty: base64-encoded string of user id and password

Return value : HTTP response status code : "200 OK" with "token=value". If sucessful, value >=0, else value:

-1: parameter error

-2: Wrong User ID or Password

-3: User ID Time Expired

- **createtoken**

Description: The initial HTTP API is to get the authorization token.

API syntax: GET HTTP/1.1 /token/createtoken?userid=xxx&password=xxxx

Parameter:

- userid: User ID
- password: User Password.

Return value : HTTP response status code : "200 OK" with "token=value". If sucessful, value >=0, else value:

-1: parameter error

-2: Wrong User ID or Password

-3: User ID Time Expired

- **destroytoken**

Description: The final HTTP API is to release the authorization token.

API syntax: GET HTTP/1.1 /token/destroytoken?token=xxxx&userid=xxx&password=xxxx

Parameter:

- token: the token returned by createtoken or createtokenbased64.
- userid: User ID.
- password: User Password.

Return value: HTTP response status code : "200 OK" with "token=value". If sucessful, value >=0, else value:

-1: parameter error

-2: Wrong User ID or Password

-3: Non-Login User ID

-4: Mismatch token for login user id

Example for login and logout

```
function login(){
 var cgi_url;
 var encrypt_str;
 var userid_pass;
 g_user_id = document.getElementById("user_id").value;
 g_password = document.getElementById("password").value;
 userid_pass=g_user_id+':'+g_password;
 encrypt_str=encode64(userid_pass);
  cgi_url =
"/token/createtokenbased64?encrpty="+escape(encrypt_str)+"&flag="+Math.random();
  xmlHttp.open("GET", cgi_url, true);
  xmlHttp.onreadystatechange = login_return;
  xmlHttp.send(null);
}

function login_return() {
  if (xmlHttp.readyState == 4)
  {
    var response = xmlHttp.responseText;
    if (response.search("-1")>0)
    {
       alert("parameter error");
    } else if (response.search("-2")>0)
    {
       alert("Wrong User ID or Password");
    } else if (response.search("-3")>0)
    {
       alert("User ID Time Expired");
    }
    else
    {
           g_token=response.slice(6,response.length);
     }
  }
}
```

```
function logout()
{
   var cgi_url;
   var confirm_msg="Logout?";
   if (confirm(confirm_msg))
   {
        g_token=find_cookie_value("token");
        g_user_id=find_cookie_value("userid");
        g_password=find_cookie_value("password");
      cgi_url =
   "/token/destroytoken?token="+escape(g_token)+"&userid="+escape(g_user_id)+
   "&password=" + escape(g_password)+"&flag="+Math.random();
         xmlHttp.open("GET", cgi_url, true);
         xmlHttp.onreadystatechange = login_out_return;
         xmlHttp.send(null);
   }
}
function login_out_return() {
   if (xmlHttp.readyState == 4)
   {
      var response = xmlHttp.responseText;
       g_token=0;
    }
}
```

3. Server APIs

Server APIs contains Channel API, Group API, Subscriber API, DVR API, System API and Billing API.

A. Channel API

- **channel_list_query**

Description: Get all channel information.

API syntax: GET HTTP/1.1 /server/channel_list_query?token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx

and "**CH=xx**\r\n**name=xxx**\r\n**src=xxx**\r\n**category=xxx**\r\n**type=xxx**\r\n**status=xxx**\r\n" for each channel, else return HTTP response status code : "404 Not Found".

Return parameter:

- **CH**: Channel No.
- **name**: Channel Name
- **src**: input stream path.
- **category**: channel category.
- **type**: live, delay,dvr, inactive.
- **status**: On/OFF

- **channel_list_add**

Description: Add a new channel.

API syntax: GET HTTP/1.1 /server/channel_list_add?token=xxx&ch_no=xxx&
ch_name=xxx&src=xxx & category=xxx&type =xxx

Parameter:

- **token**: the token returned by createtoken API.
- **ch_no**: channel no. /* ezserver will ignore it and add this channel in the last channel */
- **ch_ name**: channel name.
- **src**: input stream path.
- **category**: channel category.
- **type**: live, delay,dvr, inactive.

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

- **channel_list_update**

Description: Modify the channel information.

API syntax: GET HTTP/1.1 /server/channel_list_update? token=xxx&ch_no=xxx& ch_name=xxx&src=xxx & category=xxx&type =xxx

Parameter:

- **token**: the token returned by createtoken API.
- **ch_no**: channel no.
- **ch_ name**: channel name.
- **src**: input stream path.
- **category**: channel category.
- **type**: live, delay,dvr, inactive.

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

- **channel_list_del**

Description: Del the channel.

API syntax: GET HTTP/1.1 /server/channel_list_del? token=xxx&ch_no=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **ch_no**: channel no.

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

- **refresh_channel**

Description: Refresh all channels.

API syntax: GET HTTP/1.1 /server/ refresh_channel? token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

B. Group API

- **group_query**

Description: Get all group information.

API syntax: GET HTTP/1.1 /server/group_query?token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx

and "**No=xx**\r\n**name=xxx**\r\n**connection=xxx**\r\n**src=xxx**\r\n" for each group, else return HTTP response status code : "404 Not Found".

Return parameter:

- **No**: Group No.
- **name**: Group Name
- **connection**: max. concurrent connection
- **src**: Allowed Channels

- **group_add**

Description: Add a new group.

API syntax: GET HTTP/1.1 /server/group_add?token=xxx&group_name=xxx& group_concurrent_connection=xxx&group_src=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **group_name**: Group Name
- **group_concurrent_connection**: max. concurrent connection
- **group_src**: Allowed Channels separated by comma, ex: “1,2,3”

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

- **group_update**

Description: Modify the group information.

API syntax: GET HTTP/1.1 /server/group_update?token=xxx& group_name=xxx& group_concurrent_connection=xxx&group_src=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **group_name**: Group Name
- **group_concurrent_connection**: max. concurrent connection
- **group_src**: Allowed Channels separated by comma, ex: “1,2,3”

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

- **group_del**

Description: Delete the group information.

API syntax: GET HTTP/1.1 /server/group_del?token=xxx& group_name=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **group_name**: Group Name

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

C. Subscriber API

- **query_all_user**

Description: Get all user information.

API syntax: GET HTTP/1.1 /server/query_all_user?token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx and "**username=xxx**\r\n**password=xxx**\r\n**group=xxx**\r\n**expired_time=yy/mm/dd**\r\n **paymodel=xxx**\r\n**user_point=xxx**\r\n" for each user, else HTTP response status code : "200 OK" " with Content-Length: 0.

Return parameter:

- **username**: User Name.
- **password**: Password
- **group**: group name
- **expired_time**: channel expired time
- **paymodel**: vod model: Free, Post, Pre
- **user_point**: VOD remainder Points

- **add_user**

Description: Add a new user information.

API syntax: GET HTTP/1.1 /server/add_user?token=xxx&username=xxx&password=xxx&group=xxx&expired_time=xxx&paymodel=xxx&user_point=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **username**: User Name.
- **password**: Password
- **group**: group name
- **expired_time**: channel expired time
- **paymodel**: vod model: Free, Post, Pre
- **user_point**: VOD remainder Points

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "200 OK" with "0".

- **del_user**

Description: Delete a user information.

API syntax: GET HTTP/1.1 /server/del_user?token=xxx&username=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **username**: User Name

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "404 Not Found".

- **update_user**

Description: Update a user information.

API syntax: GET HTTP/1.1 /server/update_user?token=xxx&username=xxx&password=xxx& group=xxx&expired_time=xxx&paymodel=xxx&user_point=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **username**: User Name.
- **password**: Password
- **group**: group name
- **expired_time**: channel expired time
- **paymodel**: vod model: Free, Post, Pre
- **user_point**: VOD remainder Points

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "404 Not Found".

- **query_user_more**

Description: Get the extra information of the specified user.

API syntax: GET HTTP/1.1 /server/query_user_more?token=xxx&username=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **username**: User Name.

Return value: If sucessful, return

username=xxx\r\nsmart_phone=xxx\r\ntablet=xxx\r\ndesktop=xxx\r\ntv=xxx\r\n

first_name=xxx\r\nlast_name=xxx\r\naddress=xxx\r\ncity=xxx\r\nzip=xxx\r\ntel=xxx\r\nemail=xxx\r\n, else NULL.

Return parameter:

- **username**: User Name.
- **smart_phone**: smart phone name
- **tablet**: tablet name
- **desktop**: desktop name
- **tv**: Smart TV name
- **first_name**
- **last_name**
- **address**
- **city**
- **zip**
- **tel**
- **email**

- **save_user_more**

Description: Save the extra information of the specified user.

API syntax: GET HTTP/1.1 /server/save_user_more?token=xxx&username=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **username**: User Name.
- **smart_phone**: smart phone name
- **tablet**: tablet name
- **desktop**: desktop name
- **tv**: Smart TV name
- **first_name**
- **last_name**
- **address**
- **city**
- **zip**

Return value: If sucessful, HTTP response status code : "200 OK" with "1", else HTTP response status code : "404 Not Found".

D. DVR API

- **query_dvr_starting_time**

Description: Get the starting time of a dvr channel.

API syntax: GET HTTP/1.1 /server/ query_dvr_starting_time?token=xxx&ch_no

Parameter:

- **token**: the token returned by createtoken API.
- **ch_no**: channel no.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx and "**yyyy/MM/dd hh:mm:rr**" for the dvr channe/

E. System API

- **get_config**

Description: Get streaming ports

API syntax: GET HTTP/1.1 /get_config?token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx and "**http_port**\r\n**rtsp_port**\r\n**multicast_ip**\r\n**multicast_port**\r\n**rtmp_port**\r\n for the streaming ports.

- **save_config**

Description: Save streaming ports.

API syntax: GET HTTP/1.1/save_config?token=xxx&
httpport=xxx& rtsp_base_port==xxx&igmpip=xxx&igmpport=xxx&rtmpport=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **httpport**: http streaming port
- **rtsp_base_port**: rtsp streaming port
- **igmpip**: multicast streaming ip
- **igmport**: multicast streaming port
- **rtmpport**: rtmp streaming port

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx and "Update Port Successfully"

- **inquery_server_httpport**

Description: Get HTTP streaming port

API syntax: GET HTTP/1.1 /inquery_server_httpport?token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx and “**httpport**=xxxx\r\n for the streaming ports.

- **shutdown**

Description: Shutdown EZserver

API syntax: GET HTTP/1.1 /shutdown?token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: If sucessful, HTTP response status code : "200 OK" with "1"

F. Folder API

- **get_folder_filelist**

Description: Get all filenames in a folder

API syntax: GET HTTP/1.1 /get_user_expired_time?token=xxx?path=xxxx/xxx

Parameter:

- **token**: the token returned by createtoken API.
- **path**: the path inside ezserver folder.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxx and **"filename1**\r\n**filename2**\r\n**filename3**\r\n**"**

G. Billing API

- **get_user_expired_time**

Description: Get the expired time of a user

API syntax: GET HTTP/1.1 /get_user_expired_time?token=xxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: 10 and “**MM/dd/YYYY**” for the expired time.

- **check_user_point**

Description: Get user current point and specified movie point

API syntax: GET HTTP/1.1 /check_user_point?token=xxx&movie_name=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **movie_name**: vod movie name

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxx and **"result=xx**\r\n**user_point=xxx**\r\n**movie_point=xxx**\r\n**"**
else HTTP response status code : "404 Not Found".

Return parameter:

- **result**: 1 (User Point is enough for the Moive), -1(No User Point in Database),-2(No Movie Point in Database),-3(User Point is not enough for the Moive)

- **user_point**: User remainder Point
- **movie_point**: The needed Point for the specified movie

Movie Point is defined in /ezserver/media/eml/movie_fee.csv

```
/*
refresh time,60,min
media/videos/ts/Comedy/1.ts,100
media/videos/ts/Education/1.ts,110
media/videos/ts/Enterainment/1.ts,120
*/
```

- **charge_user_point**

Description: pay the movie point

API syntax: GET HTTP/1.1 /charge_user_point?token=xxx&movie_name=xxx

Parameter:

- **token**: the token returned by createtoken API.
- **movie_name**: vod movie name

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxx and **"result=x"** else HTTP response status code : "404 Not Found".

Return parameter:

- **result**: 1 (sucess), -1(No User Point in Database),-2(o Movie Point in Database).

Example for HTML5 Video On Demand

```
/* Index.htm */
<html>
<head>
<title>Ezhometech</title>
<link rel='stylesheet' type='text/css' href='menu.css'/>
<script src="menu.js"></script>
</head>
<body onload=login()   bgcolor="#FFFFFF" style="font-family: Verdana">
<div align="center">
  <center>
  <table border="1" cellpadding="0" cellspacing="0" width="100%">
    <tr>
      <td id=video_area width="100%">   </td>
    </tr>
  </table>
  </center>
</div>
</body>
```

/* menu.js */

```
var xmlHttp = new XMLHttpRequest();
var g_token=0;
var g_user_id;
var g_password;
var g_movie_name;

function login(){
        var cgi_url;
        var encrypt_str;
        var userid_pass;
        g_user_id = "susan";
        g_password = "1234";
        userid_pass=g_user_id+':'+g_password;
        encrypt_str=encode64(userid_pass);
        cgi_url =
"/token/createtokenbased64?encrpty="+escape(encrypt_str)+"&flag="+Math.random();
        xmlHttp.open("GET", cgi_url, true);
        xmlHttp.onreadystatechange = login_return;
        xmlHttp.send(null);
}
function login_return() {
   if (xmlHttp.readyState == 4)
   {
      var response = xmlHttp.responseText;
      if (response.search("-1")>0)
      {
          alert("parameter error");
      } else if (response.search("-2")>0)
      {
          alert("Wrong User ID or Password");
      } else if (response.search("-3")>0)
      {
          alert("User ID Time Expired");
      }
      else
      {
        g_token=response.slice(6,response.length);
        vod("/media/videos/Movie/mp4/1.mp4");
     }
   }
}

function logout()
{
        var cgi_url;
        var confirm_msg="Logout?";
        if (confirm(confirm_msg))
        {
           cgi_url =
"/token/destroytoken?token="+escape(g_token)+"&userid="+escape(g_user_id)+
"&password=" + escape(g_password)+"&flag="+Math.random();
          xmlHttp.open("GET", cgi_url, true);
          xmlHttp.onreadystatechange = login_out_return;
          xmlHttp.send(null);
        }
}
function login_out_return() {
   if (xmlHttp.readyState == 4)
   {
       var response = xmlHttp.responseText;
        g_token=0;
     }
}

function vod(move_name_str){
        var cgi_url;
        cgi_url =
"/server/check_user_point?token="+escape(g_token)+"&movie_name="+escape(move_name
_str)+ "&flag="+Math.random();
        g_movie_name= move_name_str;
        xmlHttp.open("GET", cgi_url, true);
        xmlHttp.onreadystatechange = vod_return;
        xmlHttp.send(null);
}
```

```
function vod_return () {
   if (xmlHttp.readyState == 4)
  {
     var response = xmlHttp.responseText;
      alert(response);
     if (response.search("-1")>0)
     {
         alert("No User Point in Database");
     } else if (response.search("-2")>0)
     {
       alert("No Movie Point in Database");
     } else if (response.search("-3")>0)
     {
         alert("User Point is not enough for the Moive");
     }
     else if (response.search("1")>0)
     {
      alert("User Point is enough for the Moive");
      var cgi_url;
      cgi_url =
"/server/charge_user_point?token="+escape(g_token)+"&movie_name="+escape(g_movie_n
ame)+ "&flag="+Math.random();
      xmlHttp.open("GET", cgi_url, true);
      xmlHttp.onreadystatechange = start_watch_video;
      xmlHttp.send(null);
     }
   }
}
function start_watch_video() {
  if (xmlHttp.readyState == 4)
  {
      var video_window=document.getElementById("video_area");
      var str;
      var video_path;
      video_path="http://"+location.host+g_movie_name+'?token='+g_token;
      str='<video width="640" height="480" src="'+video_path+'" controls autoplay>';
      video_window.innerHTML=str;
  }
}
function encode64(input) {
 var keyStr = "ABCDEFGHIJKLMNOP" +
              "QRSTUVWXYZabcdef" +
              "ghijklmnopqrstuv" +
              "wxyz0123456789+/" +
              "=";
    var output = "";
    var chr1, chr2, chr3 = "";
    var enc1, enc2, enc3, enc4 = "";
    var i = 0;
    do {
       chr1 = input.charCodeAt(i++);
       chr2 = input.charCodeAt(i++);
       chr3 = input.charCodeAt(i++);
       enc1 = chr1 >> 2;
       enc2 = ((chr1 & 3) << 4) | (chr2 >> 4);
       enc3 = ((chr2 & 15) << 2) | (chr3 >> 6);
       enc4 = chr3 & 63;
       output = output +
       keyStr.charAt(enc1) +
       keyStr.charAt(enc2) +
       keyStr.charAt(enc3) +
       keyStr.charAt(enc4);
       chr1 = chr2 = chr3 = "";
       enc1 = enc2 = enc3 = enc4 = "";
    } while (i < input.length);
    return output;
 }
```

4. Player APIs

Player APIs have *query_all* to get all active players.

- **query_all**

Description: Get the all active players

API syntax: GET HTTP/1.1 /player/query_all?token=xxxxxxx

Parameter:

- **token**: the token returned by createtoken API.

Return value: if successful, return HTTP response status code : "200 OK" with Content-Length: xxxx and **"username=xxx**\r\n**playername=xxx**\r\n**watching_ch**=**xxx**\r\n**player_time=xx/xx/xx xx:xx:xx**\r\n**player_ip=xxx.xxx.xxx.xxx**\r\n**country**=**xxx**\r\n**group=xxx**\r\n" for each active player, else HTTP response status code : "200 OK" " with Content-Length: 0.

Return parameter:

- **username**: subscriber name
- **playername**: User-Agent Name
- **player_time**: the starting streaming time
- **player_ip**: Player IP
- **country**: Player Location **(new parameter)**
- **group**: It decides the player to play allowed channles