TuneBlade Remote API (v0.8)

Change-list

0.1	Initial Release
0.2	- Added APIs for Streaming Modes - Added APIs for retrieving errors/logs. - Added APIs for Audio Capture Modes. - Added APIs for retrieving capture and rendering sound devices. - Added APIs for video playback. - Added APIs for activating license. - Added APIs for retrieving device specific information. - Enhanced AirPlay device information with sub state information and status of buffering. - Removed mute information from device information.
0.3	- Added an API for application exit
0.4	- Added support to connect to password protected AirPlay receivers.
0.5	- Added information on password protection
0.6	- Added support for master control
0.7	- Added support for controlling “Specific Capture Mode” - Added details on Zero-conf discovery.
0.8	- Errata in section Master Resource

General Reference

HTTP Request Methods

Method	Scenario
GET	Fetches a resource
PUT	Modifies an existing resource

HTTP Status Code

Status Code	HTTP Methods	Scenario
200	GET, PUT	Request completed successfully
400	GET,PUT	Required information was missing or malformed
401	GET, PUT	Client is not logged in or lacks sufficient privileges
404	GET, PUT	Resource not found
500	GET, PUT	An error occurred on the server

Web service Location

Base URI: <IP Address of PC where TuneBlade is running> : <Selected or automatic port number>

Zero-Conf discovery

TuneBlade remote API is discoverable using zeroconf with the following details:

Reg Type: "_http._tcp"

Name: <*MachineName*>@TuneBlade

Authorization/Authentications

The HTTP server supports basic authentication. The user name is fixed to "MYTUNEBLADE" and password can be set from the UI.

Misc. Information

Request Headers

All "PUT" requests to the web service must have the following headers set:

Header	Value
Content-Type	Application/json
Content-Length	<length of JSON>

Device Resources

Get list of all devices (AirPlay receivers) on the network, and their details.

HTTP Method	GET
URI	/devices
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	[{
	"ID": "112347690@Living Room AirPort Express",
	"Name": "Living Room Airport Express",
	"Volume": 79,
	"Status": "Connected",
	"Substate": "Streaming",
	"Buffering": false,
	"BufferingPercent": 0
	},
	{
	"ID": "1123124798@Bedroom Bose AirPlay Dock",
	"Name": "Living Room Airport Express",
	"Volume": 39,
	"Status": "Connected",

```
                "Substate": "Streaming",
                "Buffering": false,
                "BufferingPercent": 0
        },
        {
                "ID": "119076520@Kitchen JBL Soundfly",
                "Name": "Living Room Airport Express",
                "Volume": 71,
                "Status": "Connected",
                "Substate": "Streaming",
                "Buffering": false,
                "BufferingPercent": 0
        }]
```


ID: Unique identifier.

Name: The "Name" field is a user friendly name configured by the user. This should be shown as the device name in the UI. This may not be unique.

Volume: Currently set volume for a particular device. Values can be 0-100

Status: Possible values can be "Connected", "Disconnected", "Connecting", and "Disconnecting".

Substate: Possible values can be "None", "Standby", "Waiting", and "Streaming".

Buffering: Possible values can be "True" and "False".

BufferingPercent: Possible values can be 0 to 100.

Get information for a single device

HTTP Method	GET
URI	/devices/{ID of the desired device}
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	{
	"ID": "112347690@Living Room AirPort Express",
	"Name": "Living Room Airport Express",
	"Volume": 79,
	"Status": "Connected",
	"Substate": "Streaming",
	"Buffering": false,
	"BufferingPercent": 0
	}

Connect/Disconnect with a device

HTTP Method	PUT
URI	/devices/{ID}
Request Headers	Content-length Content-type

Request Body	{
	“Status”: “Connect”
	}

Response Headers	n/a
Response Body	n/a

Status: To change the status of the device to the requested status. Possible values can be “Connect” and “Disconnect”.

Note: If connection was initiated from the API and it fails, then there will be no dialog boxes from TuneBlade. The client application can get information on the failure using the ‘Log’ API as discussed later in the document.

Change volume of a device

HTTP Method	PUT
URI	/devices/{ID}
Request Headers	Content-length Content-type

Request Body	{
	“Volume”: “25”
	}

Response Headers	n/a
Response Body	n/a

Volume: Volume percentage (integer values) to be set for the desired device.

Connect to a password protected device

HTTP Method	PUT
URI	/devices/{ID}
Request Headers	Content-length Content-type
Request Body	{

	“Password”: “hello” “Status”: “Connect” }
Response Headers	n/a
Response Body	n/a

Password: Password to use while connecting to a device.

Usage Guideline:

The API client should try and connect without a password first. If the connection fails, find through the logs (see next API) whether the failure was due to missing password (AUTH_FAILED). If the failure was due to password, take the password input from the user and then use this API call. If the connection still fails due to password, this could be due to a wrong password and the user should be prompted to re-enter.

Get errors/logs for a single device

HTTP Method	GET
URI	/devices/{ID of the desired device}/**log**
Response Headers	Content-type: text/html; charset=utf-8 Content-length: <length of body>
Response Body	[<timestamp>] <Error Shorthand>

Note: The response body is in plain text format.

The response will contain error events with timestamps. The timestamp format is YYYYMMDDHHmmss.

Example: An event that took place on 18th Feb, 2014 at 12:36:09 which says the device couldn’t connect as it is busy with another stream is represented as:

[20140218123609] BUSY

Possible error codes:

BUSY: The connection failed as receiver was busy with another stream.

AUTH_FAILED: Connection failed due to missing or invalid password.

NO_RESPONSE: Connection failed as there was no response from the receiver.

IP_OR_PORT_IS_NULL: The IP address or port number could not be resolved.

SOCKET_EXCEPTION: Some connection failure such as the receiver closed the connection.

INVALID_RESPONSE: The response of the AirPlay device could not be understood.

SERVICE_UNAVILABLE: The AirPlay service on the AirPlay device is unavailable.

Streaming Mode Resource

Get currently active Streaming Mode configuration

HTTP Method	GET
URI	/StreamingMode
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	{
	“StreamingMode” : “Normal” ,
	“BufferSize” : -1
	
	}

StreamingMode: Corresponds to the four streaming modes available in TuneBlade: “RealTime”, “Normal”, “Buffered” and “Custom”.

BufferSize: Valid only when streaming mode is “Custom”, else the value should be ignored.

Set Streaming Mode configuration

HTTP Method	PUT
URI	/StreamingMode
Request Headers	Content-type: application/json Content-length: <length of body>

Request Body	{
	“StreamingMode” : “Custom” ,
	“BufferSize” : 3400
	
	}

The BufferSize configuration is not required in case the streaming mode is not “custom”.

Audio Capture Mode Resource

Get currently active Capture Mode configuration

HTTP Method	GET
URI	/CaptureMode
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	{
	“CaptureMode” : “Direct” ,
	“Virtual_IN_ID” : “{0.0.1.00000000}.{b81d5da1-32d9-4f00-b3a4}”,
	“Virtual_OUT_ID”: “{0.0.0.000000}.{b81fmg780-4567-bg56}”,
	“Specific_ID”: “{0.0.0.000000}.{b81fmg780-4567-bg56}”,
	“CanModify” : true
	}

CaptureMode: Corresponds to the four audio capture modes available in TuneBlade: “Direct” and “Virtual”.

Virtual_IN_ID: (Valid only when the capture mode is through Virtual sound devices.) The ID of the Virtual Line-In device. The ID corresponds to the ID of the sound device provided by Win32 Core Audio API MMDevice->ID.

Virtual_OUT_ID: (Valid only when the capture mode is through Virtual sound devices.) The ID of the Virtual Line-Out device. The ID corresponds to the ID of the sound device provided by Win32 Core Audio API MMDevice->ID.

Specific_ID: (Valid only when the capture mode is set to specific.) The ID of the specific device. The ID corresponds to the ID of the sound device provided by Win32 Core Audio API MMDevice->ID.

CanModify: A change to the capture mode configuration can only be done when this is set to true. This is true only when streaming is inactive. The client should first disconnect all devices from TuneBlade, and then check this value is true, before attempting to make changes to the capture mode configuration.

Make changes to the Capture Mode configuration

HTTP Method	PUT
URI	/CaptureMode
Request Headers	Content-type: application/json Content-length: <length of body>
Request Body	{

	“CaptureMode” : “Virtual” , “Virtual_IN_ID” : “{0.0.1.00000000}.{b81d5da1-32d9-4f00-b3a4}”, “Virtual_OUT_ID”: “{0.0.0.000000}.{b81fmg780-4567-bg56}”, “SPECIFIC_ID”: “{0.0.0.000000}.{b81fmg780-4567-bg56}” }

Note: The client application can get the ID for the sound devices directly from the OS through the CoreAudio MMDevice API, or can use the utility HTTP API provided by TuneBlade discussed below.

Utility API to get the list of all audio rendering devices in the system

HTTP Method	GET
URI	/listOfRenderDevices
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	[
	{
	“ID” : “{0.0.0.0000000}.{66454774-99933-999903-anfojns}”,
	“FriendlyName”: “Digital Audio (S/PDIF) (High Definition Audio”
	},
	{
	“ID” : “{0.0.0.0000000}.{9er45678d-99933-999903-anfojns}”,
	“FriendlyName”: “Cable Input (VB-Audio Virtual Cable)”
	}
	]

An optional utility function that can be used to retrieve the ID and friendly name of audio rendering devices available on the system. The same information can also be obtained from Win32 Core Audio MMDevice API.

Utility API to get the list of all audio capturing devices in the system

HTTP Method	GET
URI	/listOfCaptureDevices
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	[
	{
	“ID” : “{0.0.1.0000000}.{b81d4986-99933-999903-anfojns}”,
	“FriendlyName”: “Cable Output (VB-Audio Virtual Cable)”
	}

	]

An optional utility function that can be used to retrieve the ID and friendly name of audio capturing devices available on the system. The same information can also be obtained from Win32 Core Audio MMDevice API.

Video Player Resource

Check whether the video player is currently playing

HTTP Method	GET
URI	/videoPlayer
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	{
	“IsBusy” : false
	}

IsBusy: A value of true means the video player is alive. In order to play another video, the video player should be terminated first.

Terminating the video player application

HTTP Method	PUT
URI	/VideoPlayer
Request Headers	Content-type: application/json Content-length: <length of body>

Request Body	{
	“Stop” : “True”
	}

The client application should first check the status of the video player before attempting to play a video. If the status indicates that the video player is busy, use this API to terminate the video player before sending request to play a video.

Playing a video

HTTP Method	PUT
URI	/VideoPlayer
Request Headers	Content-type: application/json Content-length: <length of body>

Request Body	{
	“VideoLocation” : “C:\\Users\\saurabh\\desktop\\movie 01.avi”,
	“online” : “true”
	}

Note: The video location path contains forward slashes and should be escaped by preceding each forward slash by another forward slash (JSON requirement).

VideoLocation: The location specified can be of an accessible local file or of an accessible network share or a YouTube URL.

online: If the video location points to a YouTube URL, this field should be set as true.

Fine tuning audio-video synchronization

HTTP Method	PUT
URI	/VideoPlayer
Request Headers	Content-type: application/json Content-length: <length of body>

Request Body	{
	“SystemLatency” : “400”
	}

SystemLatency: TuneBlade auto-adjusts VLCs audio-delay to match the AirPlay audio. The default system latency is 200 milli-seconds. But some local system factors can make it slightly off the mark. The SystemLatency value can be used to fine-tune this. Permissible values are 100, 150, 200, 250, 300, 350, 400, 450 and 500. Values are in milli-seconds.

License Resource

Activate TuneBlade with a valid license

HTTP Method	PUT

URI	/License
Request Headers	Content-type: application/json Content-length: <length of body>

Request Body	{
	“Key” : “7677NFGDGD9095838383JFJFJF”
	“Key ID” : “xyz@xyz.com”
	}

A valid license information can be provided through this API to unlock TuneBlade.

Functions Resource

Shutdown TuneBlade

HTTP Method	PUT
URI	/Functions
Request Headers	Content-type: application/json Content-length: <length of body>

Request Body	{
	“Exit” : “Normal”
	}

Exit: Two possible values: ‘Normal’ and ‘Force’. The ‘Normal’ exit is equivalent to pressing the exit button on TuneBlade UI. The shutdown happens gracefully by disconnecting the streams, releasing memory etc., and may take a few seconds. The ‘Force’ kills TuneBlade immediately. This is useful in a rare scenario, if the normal shutdown takes longer than expected.

Master Resource

Check the status of master control panel

HTTP Method	GET
URI	/master
Response Headers	Content-type: application/json Content-length: <length of body>

Response Body	{
	“IsEnabled” : false,
	“Status” : “Connected”,
	“Volume”: 5
	}

Status: Possible values: Connected, Connecting, Disconnecting, Disconnected.

Change the status of master

HTTP Method	PUT
URI	/master
Request Headers	Content-type: application/json Content-length: <length of body>

Request Body	{
	“Status” : “Connect”,
	“Volume”: 5
	
	}