SoundBox 2.9 API v7

Revision History

24 Dec 2015	1.0	First release	APC
29 Dec 2015	1.1	Note about timer pre-delay	APC
31 Dec 2015	1.2	Addition of API version error	APC
		Addition of 'runningIndex' property of timerInfo	APC
		Addition of 'lengthMillisecs' property of songInfo	APC
		Addition of bell API	APC
		Addition of API version address	APC
01 Jan 2016	1.3	Note about sample Windows app	APC
15 Jan 2016	1.4	Note about 'grace second'	APC
09 May 2016	1.5	Added API code	APC
		Added system API	APC
		Added media API	APC
12 May 2016	1.6	Added support for thumbnails in media API following Daniel's review (10 May)	APC
		Added sessionId and api control flags in system API	APC
		Refactored media API	APC
	1.7	Added "jumping into a slideshow"	APC
14 May 2016	1.8	Added media status API	APC
17 May 2016	1.9	Added query string api code option	APC
19 May 2016	1.10	Added UDP broadcast response	APC
27 May 2016	1.11	Added apiCodeRequired property in system API	APC
30 May 2016	1.12	Minor update to UDP spec	APC
26 June 2016	1.13	Added isApplicable property to Timer object	APC
21 July 2016	1.14	Added section on Media Listing Codes and sorting of media items	APC
26 July 2016	1.15	Added support for video thumbnails and 512x512 thumbnail size	APC
		Added support for starting a slideshow at an offset	APC
		Added support for jpg thumbnail format	APC
31 July 2016	1.16	Updated screenshots for SoundBox v3	APC
10 Sep 2016	1.17	Added control of song numbers and background music	APC
21 Oct 2016	1.18	Added note regarding internalTabName in Timers API	APC
26 Oct 2016	1.19	Revised note regarding internalTabName in Timers API	APC
28 Oct 2016	1.20	Added Event Notifications (api v7)	APC
29 Oct 2016	1.21	Added "duration" to mediaInfo object	APC
		Added "duration" and "position" to mediaStatus objects	APC
30 Oct 2016	1.22	Additional event notifications	APC
05 Nov 2016	1.23	Additional event notifications	APC

Contents

1. Introduction

This specification describes a simple web API that can be used with SoundBox software (http://cv8.org.uk/soundbox). The API allows 3rd party application developers to retrieve selected data from SoundBox while it's executing, and to control some aspects of its execution.

1.1 Server

The SoundBox application itself acts as the http server so the API can only be used when SoundBox is running.

1.2 Scope

The API currently covers 4 main areas – timers, media, songs and background music. The *timers* API provides information on each of the talk timers and allows you to switch between each of the timer's 3 states (ready, running and stopped). The *media* API lists available media items such as videos and images, and provides playback control. The *songs* API provides information on each of the 3 songs queued up in SoundBox for the meeting, allows you to specify song numbers, and allows you to start a song. The *background music* API provides status information and allows you to start and stop the background music. Finally, there is an *event notification* scheme that allows you to subscribe to common SoundBox events such as showing an image or starting a timer.

The main rationale for developing the API is the desire to more easily distribute the SoundBox functions among operators at the KH. Currently a single operator may be responsible for microphone mixing, playing songs, media presentation, management of a conference system, timing, monitoring security cameras, etc. If remote mobile apps can be used to manage the SoundBox timers, or play the songs and media, then others may be enlisted to help. A secondary motivation is to ease integration with other IT systems such as electronic information boards where it might be useful to display timers and songs.

2. API Description

2.1 Base URI

The base URI is as follows:

```
http://soundbox_machine:8095/api/v7
```

Please replace "soundbox_machine" with the name or IP address of the PC on which SoundBox is running. You must be able to connect to the machine from your client device, e.g. over a WiFi network.

Note that the port (8095) is currently hard-coded in SoundBox but may be configurable in future versions. The API version designation allows the interface to be modified in future revisions whilst retaining backwards compatibility. The idea is that if you write client code against a particular version of the API it will still work when subsequent versions are released.

Getting the Supported API Versions

You can retrieve the lowest and highest API versions supported by a SoundBox installation by issuing a GET request at the following address:

```
http://soundbox_machine:8095/api/
```

SoundBox responds with JSON structure in the following format:

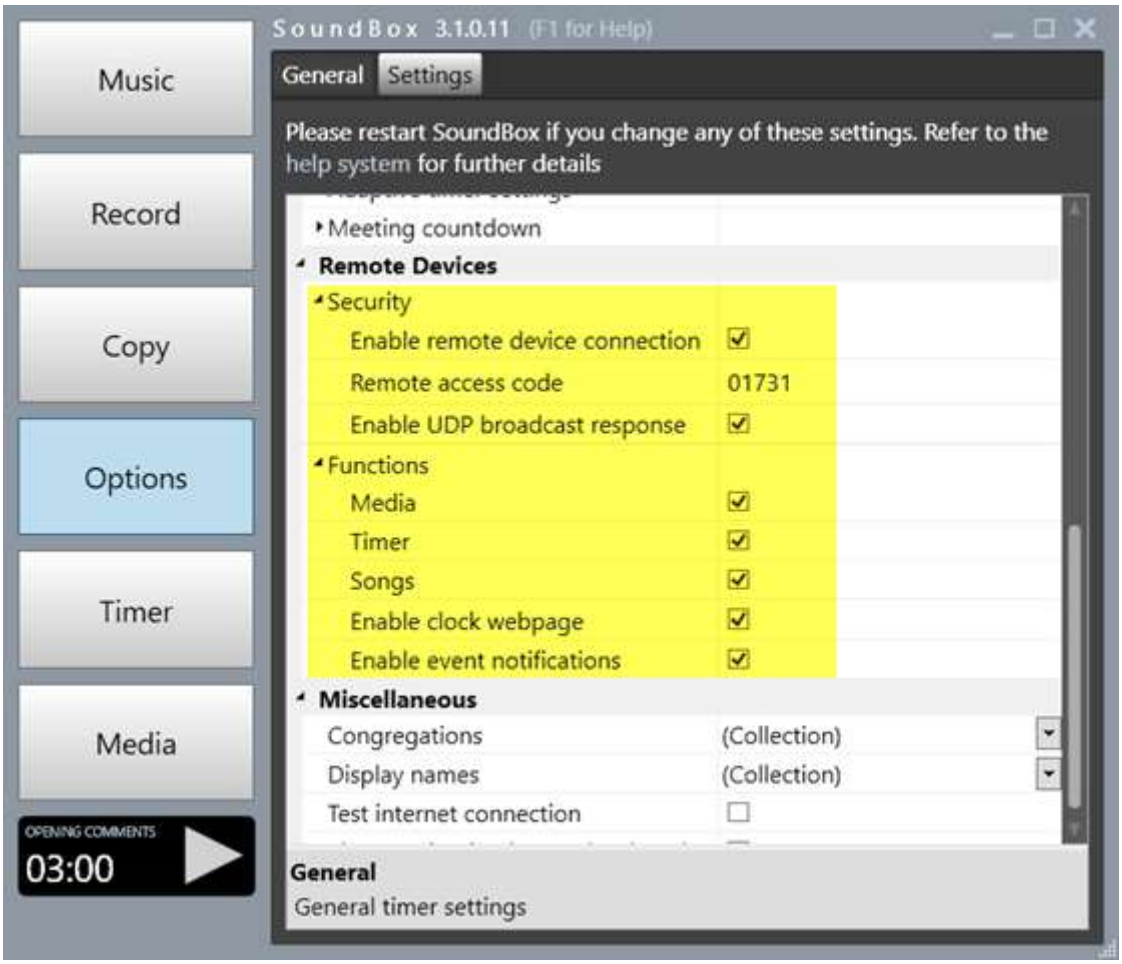
```
{
  "lowVersion":[lowest version],
  "highVersion":[highest version]
}
```

The elements are described below

- **lowVersion** – this is the lowest supported API version. Of course, we will try to retain backwards compatibility with all API versions.
- **highVersion** – this is the highest supported API version.

Unfortunately, this information is only available starting at version 2 of the API. If you issue the above request and you get a 503 http status code (service unavailable) then SoundBox probably supports v1 of the API only.

You must enable the SoundBox web API (in Options, Settings, Remote Devices) and enable any specific APIs that are required by your client application as shown below:



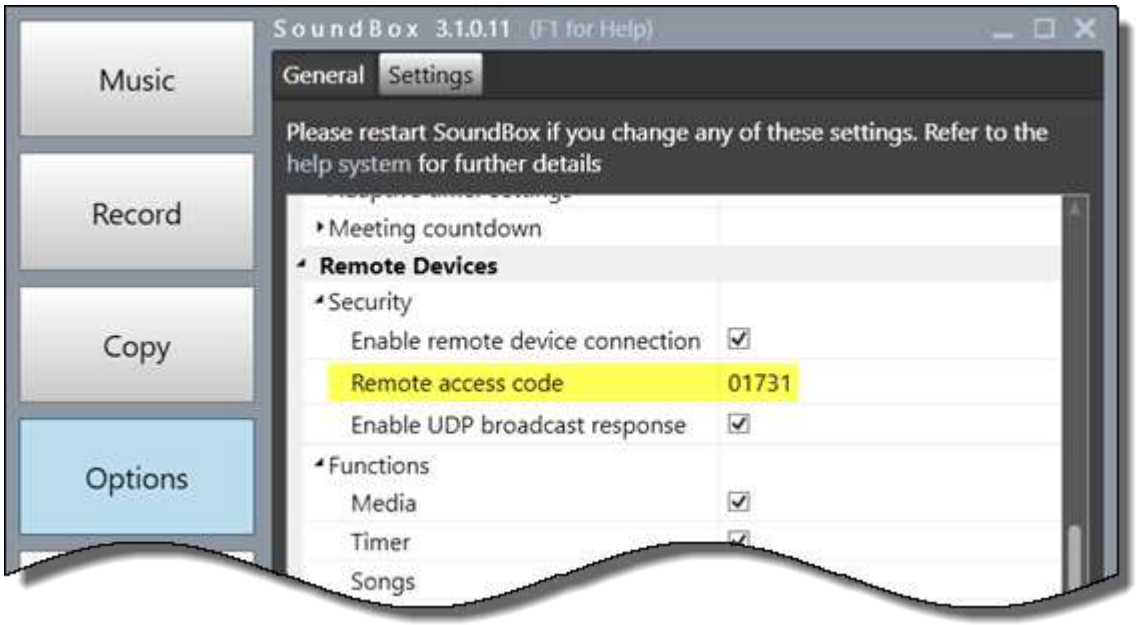
The "Enable clock webpage" setting is used to control a single web page served by SoundBox. This can be useful to test connectivity between your mobile device and SoundBox. Enable the setting, open a web browser on your client device and navigate to the following clock URI:

```
http://soundbox_machine:8095/index
```

The web browser should display the time of day. If you then start a timer in SoundBox, the browser should display the timer countdown value.

2.2 Security

From v3 of the API (available in SoundBox v2.9.0.172), SoundBox implements an optional security scheme which, when enabled, requires a "remote access code" to be passed as a request header or as a query string parameter. To enable this feature, you must specify a code in the SoundBox Options, Settings, Remote Devices section as shown below:



The value can be anything you like and is not restricted to digits. However, if it needs to be entered on a mobile device then you may want to keep it short and use digits.

Please remember that unless communication between your clients and SoundBox is secured using SSL (https) then traffic will be in plain text and the remote access code will be available to a snooper. However, given the controlled environment of a WiFi network in a Kingdom Hall this is generally considered low risk.

If you decide to use a remote access code, when constructing client-side requests you must either add a header with the key "ApiCode" or pass a query string parameter named "ApiCode". The value should be as specified above in the SoundBox Options. See below for an example of the header:

```
GET /api HTTP/1.1
Host: localhost:8095
ApiCode: 8273
Content-Type: application/json
```

Passing the remote access code by query string is illustrated below:

```
http://soundbox_machine:8095/api/v7/media?ApiCode=8273
```

The following API calls do *not* require the code:

```
http://soundbox_machine:8095/api/
```

```
http://soundbox_machine:8095/api/v7/system
```

2.3 System

The system API (from v3) provides access to some basic SoundBox information and has the following URI:

```
http://soundbox_machine:8095/api/v7/system
```

There is no need to provide a remote access code in your request. A GET request sent to the above address will return a single object in the following JSON format:

```
{
  "machineName": "[name of the PC]",
  "accountName": "[name of the current windows account]",
  "soundBoxVersion": "[SoundBox software version]",
  "apiVersion": {
    "lowVersion": [lowest API version],
    "highVersion": [highest API version]
  },
  "congregationName": [congregation name],
  "culture": {
    "name": "[culture string]",
    "isoCode2": "[ISO 2-character language code]",
    "isoCode3": "[ISO 3-character language code]"
  },
  "internet": {
    "isChecked": [true/false],
    "connected": [true/false/null]
  },
  "workingSet": [working set memory in bytes]
  "sessionId": "[SoundBox session Id string]"
  "timerApiEnabled": [true/false],
  "songsApiEnabled": [true/false],
  "mediaApiEnabled": [true/false],
  "eventNotificationsEnabled": [true/false],
  "apiCodeRequired": [true/false]
}
```

Note that "eventNotificationsEnabled" is only available from v7 of the API onwards.

A typical response is shown below:

```
{
  "machineName": "SoundPC",
  "accountName": "Allington",
  "soundBoxVersion": "2.9.0.171",
  "apiVersion": {
    "lowVersion": 1,
    "highVersion": 3
  },
  "congregationName": "",
  "culture": {
    "name": "en-GB",
    "isoCode2": "en",
    "isoCode3": "eng"
  },
  "internet": {
    "isChecked": true,
    "connected": true
  },
  "workingSet": 100184064,
  "sessionId": "2b44993d-5539-483d-9c68-9ef005177d12"
  "timerApiEnabled": true,
  "songsApiEnabled": true,
  "mediaApiEnabled": false,
  "eventNotificationsEnabled": true,
  "apiCodeRequired": true
}
```


The elements are described below

- **machineName** – NetBIOS name of the machine on which SoundBox is running
- **accountName** – current logged on user account name
- **soundBoxVersion** – SoundBox version, e.g. "2.9.0.157"
- **apiVersion.lowVersion** – lowest supported API version
- **apiVersion.highVersion** – highest supported API version
- **congregationName** – name of congregation (if specified)
- **culture.name** – name of the current culture, e.g. "en-GB"
- **culture.isoCode2** – ISO 2 character language code
- **culture.isoCode3** – ISO 3 character language code
- **internet.isChecked** – a Boolean value to denote whether SoundBox monitors internet connectivity
- **internet.connected** – a nullable Boolean value to indicate whether the PC is connected to the internet (true) or not (false), or has not yet checked (null)
- **workingSet** – the amount of physical memory mapped to the process context (in bytes)
- **sessionId** – a unique identifier that corresponds to an instance of SoundBox. You can use this to determine whether SoundBox has been closed and re-opened since your last request
- **timerApiEnabled** – a Boolean value indicating whether the timer can be controlled via the API
- **songsApiEnabled** – a Boolean value indicating whether the songs can be controlled via the API

- **mediaApiEnabled** – a Boolean value indicating whether media can be controlled via the API
- **eventNotificationsEnabled** – a Boolean value indicating whether it is possible to subscribe to event notifications (see the "Event Notifications" section for more information). This property is only available from v7 of the API onwards
- **apiCodeRequired** – a Boolean value indicating whether use of the API requires a code.

2.4 Timers

The timers API has the following base URI:

```
http://soundbox_machine:8095/api/v7/timers
```

Getting the Timer Collection

A GET request sent to the above address will return a collection of timer objects in the following JSON format. Note that "internalTabName" is only available from API v3 onwards, and "isApplicable" from v4 onwards:

```
{
  "timerInfo":
    [
      {
        "internalName":"[internal name]",
        "localisedTitle":"[localised title of timer]",
        "localisedTabName":"[localised tab title]",
        "internalTabName":"[internal tab title]",
        "status":[status value],
        "index":[index value],
        "isEnabled":[true/false],
        "isApplicable":[true/false],
        "isStudentTalk":[true/false],
        "plannedAllocationSecs":[planned allocation in secs],
        "actualAllocationSecs":[actual allocation in secs],
        "elapsedMillisecs":[elapsed millisecs],
        "displayedMins":[number of mins displayed on timer],
        "displayedSecs":[number of seconds displayed on timer],
        "runningIndex":[index of currently running timer]
      },
      {
        ...
      }
    ]
}
```

A typical example is shown below (truncated to show only 1 timer):

```
{
  "timerInfo":
    [
      {
        "internalName":"OpeningComments",
        "localisedTitle":"OPENING COMMENTS",
        "localisedTabName":"Treasures",
        "internalTabName":"Treasures",
        "status":0,
        "index":0,
        "isEnabled":true,
        "isApplicable":true,
        "isStudentTalk":false,
        "plannedAllocationSecs":180,
        "actualAllocationSecs":180,
        "elapsedMillisecs":0
        "displayedMins":0,
        "displayedSecs":0,
        "runningIndex":-1
      },
      {
        ...
      }
    ]
}
```

The elements are described below

- **internalName** – the name given to the timer internally in SoundBox (this doesn't change and so can be used within client code, but see also "index" below)
- **localisedTitle** – the localised name given to the timer (i.e. one that changes for each language version)
- **localisedTabName** – the localised name that appears on the timer's tab (there are several timer tabs in SoundBox; each one covering a meeting or part of a meeting)
- **internalTabName** (from v3) – the internal name of the timer's tab. "Treasures", "Ministry", "Living", "Sunday" or "Miscellaneous". Note that the "Sunday" tab has recently been labelled "Weekend", and the "Miscellaneous" tab as "Custom" in the latest version of SoundBox, but the API continues to use "Sunday" and "Miscellaneous" as the internal tab names for the sake of continuity.
- **status** – the status of the timer (0 = Ready, 1 = Running, 2 = Stopped)
- **index** - a unique integer value for the timer. This value also denotes the timer's zero-based ordinal position. The SoundBox API guarantees that the timer elements will be returned in ascending order sorted by index
- **isEnabled** – a Boolean value to denote whether the timer is enabled (i.e. can be transitioned). This reflects the state of the timer in the SoundBox user interface, i.e. it is false when any timer is running
- **isApplicable** (from v4) – a Boolean value to denote if a timer is applicable given the current date and user settings. For example, on the first midweek meeting of the month the timer

labelled "This week's presentations" is normally applicable and the 3 student talk timers are not

- **isStudentTalk** - a Boolean value to denote whether the timer is associated with a student talk
- **plannedAllocationSecs** – the time (in seconds) that is allocated to the timer
- **actualAllocationSecs** – the actual time (in seconds) that is allocated to the timer. This is usually the same as plannedAllocationSecs but may be different if SoundBox's *adaptive timing* mode is enabled
- **elapsedMillisecs** – the current time elapsed
- **displayedMins** – the current 'minutes' value displayed on the timer
- **displayedSecs** – the current 'seconds' value displayed on the timer
- **runningIndex** (from API v2) – the index (zero-based) of the currently running timer (or -1 if no timers are running)

Getting an Individual Timer

A GET request sent to the following URL retrieves information about the specified timer:

```
http://soundbox_machine:8095/api/v7/timers/[timer index]
```

The "timer index" is an integer value corresponding to the index field of the timer (see "Getting the Timer Collection" above). In practice, the timer index values range from 0 – n (where n = number of timers minus 1), and this is unlikely to change because *index* also represents the ordinal position of the timer.

A sample JSON response is shown below:

```
{
  "internalName":"Reading",
  "localisedTitle":"BIBLE READING",
  "localisedTabName":"Treasures",
  "internalTabName":"Treasures",
  "status":0,
  "index":3,
  "isEnabled":true,
  "isApplicable":true,
  "isStudentTalk":true,
  "plannedAllocationSecs":240,
  "actualAllocationSecs":240,
  "elapsedMillisecs":0
  "displayedMins":0,
  "displayedSecs":0
  "runningIndex":-1
}
```

Starting, Stopping and Resetting a Timer

A POST request sent to the following URI transitions the specified timer to its next state:

```
http://soundbox_machine:8095/api/v7/timers/[timer index]
```

Timers have 3 states: Ready, Running and Stopped. The states are strictly ordered and you can transition a timer from Stopped to Ready (which is equivalent to using the *Reset* command in the user interface).

SoundBox provides a JSON response consisting of the new timer information.

Note – SoundBox automatically stops all timers at 10 mins overtime.

Sounding the Timer Bell

The timer bell can be sounded (from API v2) by posting to the following URL:

```
http://soundbox_machine:8095/api/v7/bell
```

SoundBox returns a simple JSON object with the following structure:

```
{
  "success":[true/false]
}
```

2.5 Songs

The songs API has the following base URI:

```
http://soundbox_machine:8095/api/v7/songs
```

Getting a Collection of Song Controls

A GET request sent to the above address will return a collection of song control objects in the following JSON format:

```
{
  "songInfo":
    [
      {
        "index":[index value],
        "internalName":"[internal name of the song control]",
        "localisedName":"[localised name of the song control]",
        "title":"[the official localised song title]",
        "status":"[the status of the song control]",
        "isEnabled":[true/false],
        "songNumber":[the official song number],
        "lengthMillisecs":[the song duration]
      },
      {
        ...
      }
    ]
}
```

A typical example is shown below (truncated to show only 1 song control):

```
{
  "songInfo":
    [
      {
        "index":0,
        "internalName":"Opening",
        "localisedName":"OPENING SONG",
        "title":"50 - The Divine Pattern of Love",
        "status":0,
        "isEnabled":true,
        "songNumber":50,
        "lengthMillisecs":184539
      },
      {
        ...
      }
    ]
}
```

The elements are described below:

- **index** - a unique integer value for the song control. 0 = opening song, 1 = middle song, 2 = closing song. The SoundBox API guarantees that the song control elements will be returned in ascending sorted by index
- **internalName** – the name given to the song control internally in SoundBox (this doesn't change and so can be used within client code, but see also "index" above)
- **localisedName** – the localised name given to the song control (i.e. one that changes for each language version)
- **title** – the official title of the song (language-specific)
- **status** – the status of the song control (0 = ready to play, 1 = playing, 2 = empty, i.e. no song number)
- **isEnabled** – a Boolean value to denote whether the song control is enabled (i.e. can be played)
- **songNumber** – the official song number (between 1 and 145 at time of writing). Note that if the song field is empty the returned songNumber value is 0
- **lengthMillisecs** (from API v2) – the duration of the song in milliseconds

Getting an Individual Song Control

A GET request sent to the following URL retrieves information about the specified song control:

`http://soundbox_machine:8095/api/v7/songs/[control index]`

The song "control index" is an integer value corresponding to the index field of the song control (see "Getting a Collection of Song Controls" above). In the current version of the API there are 3 song controls and they have indexes 0, 1 and 2 (but please rely on the values retrieved from the collection instead of hard-coding these indexes).

A sample JSON response is shown below:

```
{
  "songInfo":
    [
      {
        "index":2,
        "internalName":"Closing",
        "localisedName":"CLOSING SONG",
        "title":"133 - Seek God for Your Deliverance",
        "status":0,
        "isEnabled":true,
        "songNumber":133,
        "lengthMillisecs":145555
      },
      {
        ...
      }
    ]
}
```

Note that if the song field in SoundBox is empty the returned songNumber value is 0.

Specify a Song Number (from API v6 only)

A POST request sent to the following URI will change the number of the specified song:

`http://soundbox_machine:8095/api/v7/songs/[control index]/[number]`

SoundBox provides a JSON response consisting of the new song control information.

For example, to change the opening song to song 123 you would POST to this URI:

`http://soundbox_machine:8095/api/v7/songs/0/123`

You can clear one of the song controls by specifying song 0. So a POST to the following URI would clear the middle song:

`http://soundbox_machine:8095/api/v7/songs/1/0`

Playing and Stopping a Song

A POST request sent to the following URI will toggle the status of the specified song (between playing and stopped):

`http://soundbox_machine:8095/api/v7/songs/[control index]`

SoundBox provides a JSON response consisting of the new song control information.

A client would not usually stop a song; it is usual for SoundBox to play a song to completion when it then stops automatically. You can detect when a song has finished by polling the song control and checking the status value (the *lengthMillisecs* property may be useful in this respect).

2.6 Background Music

The background music API (available from v6 of the API) has the following base URI:

```
http://soundbox_machine:8095/api/v7/background-music
```

Getting the Status of Background Music

A GET request sent to the following URI will return status information about the background music:

```
http://soundbox_machine:8095/api/v7/background-music
```

A sample JSON response is shown below:

```
{
  "status": 0,
  "autoStopEnabled": true,
  "autoStopped": false
}
```

The elements are described below:

- **status** – the status of background music. 0 = not playing, 1 = playing, 2 = not available (e.g. if a song is being played)
- **autoStopEnabled** - a Boolean value to denote whether background music is set to stop automatically before the start of the meeting
- **autoStopped** - a Boolean value to indicate if the music has stopped automatically (via the auto stop background music option)

Playing and Stopping Background Music

A POST request sent to the following URI will start or stop the background music (depending on the JSON specified in the body of the request):

```
http://soundbox_machine:8095/api/v7/background-music
```

A typical body is shown below:

```
{
  "action": "play"
}
```

The **action** element specifies the operation to perform. Possible values for action are described below:

- **play** – starts the background music
- **stop** – stops the background music

SoundBox responds with success (http status code 200) if all is well. Note that when background music is stopped it takes a few seconds to fade out. During this time the background music status is still "playing".

2.7 Media

The media API has the following root URI:

```
http://soundbox_machine:8095/api/v7/media
```

Getting the Media List

A GET request at the media URI gets a list of all media items. A sample JSON response is shown below:

```
{
  "mediaInfo": [
    {
      "id": "5015102",
      "title": "M1-010 A sister meditates on what she...",
      "type": "Image"
    },
    {
      "id": "5015101",
      "title": "M3-010 pk009—Jehovah . . . Created All Things",
      "type": "Video"
      "duration": 156000
    },
    {
      "id": "5015103",
      "title": "M3-020 Martha",
      "type": "Image"
    },
    {
      "id": "5015104",
      "title": "M3-030 Mary sitting at Jesus' feet listening...",
      "type": "Image"
    },
    {
      "id": "5015105",
      "title": "JW Broadcasting",
      "type": "Url"
    },
    {
      "id": "5015106",
      "title": "JW.org website",
      "type": "Url"
    },
    {
      "id": "5015107",
      "title": "Watchtower online library",
      "type": "Url"
    }
  ]
}
```

Note that the response is limited to a maximum of 100 media items to assist system performance. SoundBox is designed in such a way as to encourage users to keep the media lists short so the restriction should not pose a difficulty.

The elements are described below:

- **id –** a string value that uniquely identifies the media item during a SoundBox session

- **title** – the media title as it appears in the SoundBox media list (but see below under Media Listing Codes)
- **type** – the type of media represented. Possible values are "Video", "Image", "Url", "Pdf", "Audio", and "Slideshow"
- **duration** – the duration of the media in milliseconds (only applicable to items of media type "Video" and "Audio"). Only available from API v7 onwards

Note that all "id" values are valid for the current SoundBox session only – they should not be persisted on the client. You can determine the current session Id using the "system" API.

Note that "duration" is only available from v7 of the API onwards.

Media Listing Codes

SoundBox has a media service function which is optionally used to automatically download media files (as opposed to manually finding the relevant resources and adding them to the SoundBox media folders). When this service is used, the "title" attribute of the mediaInfo object begins with a *Media Listing Code*.

The format of the Listing Code is as follows:

MN-XYZ

Where M is a character denoting whether the meeting is the midweek one ('M') or the weekend ne ('W'). N is a single digit referring to the meeting part (on the weekend this is 1 for the Public Talk and 2 for Watchtower; and midweek it is 1, 2 or 3 for the standard meeting sections). Finally, XYZ are digits and designate the order of the media item in the meeting. These numbers are typically issued in increments of 10 to allow users to manually insert media in the gaps.

Some examples of media listing codes:

M1-010

A media item from the first section ("Treasures…") of the midweek meeting

W2-030

A media item (typically an image) in the Watchtower study

The Media Listing Code is not shown in the SoundBox application; it is stripped from the title to make the user-interface more attractive. You can choose to do a similar thing if you need to display the media title to your users. Since the codes are always 6 characters in length and are followed by a space you can safely strip the first 7 characters from the title once you have determined that a listing code is present.

Ordering

The mediaInfo items returned in the GET request are sorted by listing code and then by title. Items with a listing code always appear before those without. The title sort order respects the current culture settings and is case-insensitive.

Slideshows

Where an item is of type "Slideshow" you can get a list of media items it contains using the following URI:

```
http://soundbox_machine:8095/api/v7/media/[slideshow id]
```

A GET request at the following URI will list all the images in the specified slideshow:

```
http://soundbox_machine:8095/api/v7/media/1314111
```

Thumbnail Images

You can retrieve thumbnail images for all Image, Video and Slideshow items using one of the following URIs:

```
http://soundbox_machine:8095/api/v7/media/thumbs64/[media id]
```

```
http://soundbox_machine:8095/api/v7/media/thumbs128/[media id]
```

```
http://soundbox_machine:8095/api/v7/media/thumbs256/[media id]
```

In v5 of the API, support for 512x512 was added:

```
http://soundbox_machine:8095/api/v7/media/thumbs512/[media id]
```

The images have a maximum size of 64x64, 128x128, 256x256 and 512x512 pixels respectively – choose the one that best meets the requirements of your application. Thumbnails retain the aspect ratio of the original image (they are not distorted).

Thumbnails are available only for Images in v4 of the API. In v5 onwards, Video thumbnails are also available.

Note that the thumbnail format is png up to and including API version 4. From v5 of the API, the thumbnail format is jpg.

Controlling Media Playback

A POST request sent to the media URI is used to control the playback of items. The identification of the media item and the control operation are specified in JSON format in the body of the POST request as shown below:

```
{
  "id": "5584109",
  "action": "play"
}
```

The **action** element specifies the operation to perform on it.

Possible values for action are described below:

- **play** – plays a video or audio clip or shows an image, web site, etc
- **stop** – stops a video or audio clip or hides an image, web site, etc
- **pause** – pauses a video clip
- **next** – displays the next photo in a slideshow
- **prev** – displays the previous photo in a slideshow
- **blank** – displays a blank slide in a slideshow

SoundBox responds with success (http status code 200) if all is well. Additionally, an object is returned which in some circumstances (notably slideshow navigation) can provide extra

information. For example, the following response might be obtained while navigating a slideshow using the "next" action:

```
{
  "id": "3422110",
  "action": "next",
  "slideIndex": 3,
  "slideId": "3422121"
}
```

- **slideIndex** refers to the zero-based index into the slideshow
- **slideId** is the mediaId of the actual image currently being shown

If there is an error condition SoundBox provides an error code as described under "Error Handling" below.

Jumping into a Slideshow

With slideshows, you can also use the "play" action to jump to a specified slide. Simply add a "slideIndex" value to the body of your request as illustrated below:

```
{
  "id": "9436110",
  "slideIndex": 10,
  "action": "play"
}
```

In the above example, "id" is the slideshow media id, and slideIndex is the zero-based slide number in the sequence. Once you have 'jumped' into a slideshow using the above syntax you can continue by using "next", "prev" and "stop", or you can jump to another slide in the sequence.

Up to v4 of the API you can use this technique only during an active slideshow (i.e. a slideshow must always start at the first slide). From v5 of the API, it can also be used to start a slideshow at an offset.

SoundBox Mode

When the API is used to control media, SoundBox displays a notification message alerting the operator as shown below:

In this situation, it is possible for the operator to control media at the same time, requiring cooperation to make it worth smoothly!

2.8 Media Status

The media status API has the following root URI:

http://soundbox_machine:8095/api/v7/media-status

The API is used to determine the status of current SoundBox media playback and to make generic control operations.

Getting Media Status

A GET request to the above URI retrieves the current status object in the following form:

```
{
  "status": [current status]
  "id": [id of the current media item],
  "title": [title of the current item],
  "type": [type of the current item],
  "slideId": [id of the current slide in a slideshow],
  "slideIndex": [index of the current slide in a slideshow],
  "duration": [duration of media in millisecs],
  "position": [position of media in millisecs]
}
```

An example is shown below:

```
{
  "status": "Inactive",
  "id": "9031101",
  "title": "Brothers—Reach Out for a Fine Work",
  "type": "Video",
  "duration": 123000,
  "position": 4560,
}
```

The elements are described below:

- **status** – The current status of media playback. Possible values are "Inactive", Active" and "Paused" (where "Paused" is only applicable to video items)
- **id** – The id of the current media item. This is omitted if no media item is currently selected in SoundBox
- **title** – The title of the current media item (if one is selected)
- **type** – The type of the current media item if one is selected. Possible values are "Video", "Image", "Url", "Pdf", "Audio", and "Slideshow"
- **slideId** – the id of the current slide in a slideshow (only applicable to items of type "Slideshow")
- **slideIndex** – the zero-based index of the current slide in a slideshow (only applicable to items of type "Slideshow")
- **duration** – the duration of the media in milliseconds (only applicable to items of type "Video" and "Audio"). Only available from API v7 onwards
- **position** – the current playback position of the media in milliseconds (only applicable to items of type "Video" and "Audio"). Only available from API v7 onwards

Notes:

- "duration" and "position" are only available from v7 of the API onwards.
- The duration and position may not be available until the media has been playing for a second or so.

Changing Media Status

You can also *control* the current media item using the media status API. This is achieved by issuing a POST request to the URI specifying the relevant action. For example, the following JSON body starts to play the current item:

```
{
    "action": "play"
}
```

The API responds with the media status as reported by SoundBox following the specified action.

The following example shows how to stop the current media item (without having to know its identity):

```
{
    "action": "stop"
}
```

Note that the "media status" API operates on the *current media item*. However, if you want more control over a specific media item then please use the "media" API described in the previous section.

2.9 Local Date / Time

You can get the current SoundBox date/time using this URI:

```
http://soundbox_machine:8095/api/v7/datetime
```

A sample JSON response is shown below:

```
{
  "year":2015,
  "month":12,
  "day":17,
  "hour":15,
  "min":11,
  "second":51
}
```

The fields are self-explanatory. If you want to display the "current time" in your client application, then it may be best to read the SoundBox local time at application startup and then store the difference between that and your device's local time - thus allowing you to display the SoundBox time whenever needed. Note that the latency of your network will affect the accuracy of the response value.

2.10 Error Handling

Standard http error codes are used where possible. A successful request meets with a response having an "ok" status code (200), but suppose you attempt to GET a song that doesn't exist, e.g. by using the following address:

```
http://soundbox_machine:8095/api/v7/songs/3
```

In this case the SoundBox API returns an http error code 404 ("not found") because there are only 3 songs (with indexes 0 – 2).

The SoundBox API may also return a JSON error object in the following format if more information is available:

```
{
  "errorCode":[An internal error code],
  "errorMessage":"[Description of the error]",
  "conflictingId":"[id of any conflicting item]"
}
```

The **conflictingId** element is not always present but can provide additional context about the error condition. For example, there follows a typical error object returned if you try to play a video when an image (with id = 1087107) is currently being displayed:

```
{
  "errorCode": 1147,
  "errorMessage": "Media busy",
  "conflictingId": "1087107"
}
```

Error Code List

The following SoundBox error codes are used (returned in the JSON error object described above):

Code	Message and Description	Http Status Code
0	"Success"	200 (ok)
1128	"Song control does not exist" – you have specified a song control index that doesn't exist. For example, http://soundbox_machine:8095/api/v7/songs/3	404 (not found)
1129	"Timer does not exist" – you have specified a timer index that doesn't exist. For example, http://soundbox_machine:8095/api/v7/timers/100	404 (not found)
1130	"Malformed URI" – you have specified a URI with too many segments. For example http://soundbox_machine:8095/api/v7/timers/1/foo	400 (bad request)
1131	"Malformed URI" – you have specified a URI with too few segments. For example http://soundbox_machine:8095/api/v7/	400 (bad request)
1132	"Could not identify timer" – it wasn't possible to identify a timer value in the URI. For example http://soundbox_machine:8095/api/v7/timers/abc	400 (bad request)

1133	"Could not identify song control" – it wasn't possible to identify a song control value in the URI. For example http://soundbox_machine:8095/api/v7/songs/abc	400 (bad request)
1134	"Malformed URI" – you have specified a prefix that SoundBox cannot handle. For example http://soundbox_machine:8095/api/v7/foobar/1	400 (bad request)
1135	"Wrong http method used" – you have used an http method that is not supported by the target resource. For example you can issue a POST request to the root of the timers resource	405 (method not allowed)
1136	"Could not play/stop song" – you have successfully issued a request to toggle the status of a song but SoundBox cannot fulfil the request (e.g. because another song is playing)	409 (conflict)
1137	"Could not transition timer" – you have successfully issued a request to transition a timer but SoundBox cannot fulfil the request (e.g. because another timer is running)	409 (conflict)
1138	"API version not supported" – you are trying to use an API version that is not supported by this version of SoundBox	400 (bad request)
1139	"Not available in selected API version" – you are trying to use part of the API that is not available in the API version that you specified in the URI	400 (bad request)
1140	"Invalid API code" – you have not specified a valid remote access code in the header of your request	401 (unauthorised)
1141	"No media target display" – the media target display is not set	409 (conflict)
1142	"Invalid media command" – the specified action is unknown or not applicable in the current context	409 (conflict)
1143	"Media not found" – the media item was not found	404 (not found)
1144	"Unknown media tab name" – the specified media tab name is unknown	400 (bad request)
1145	"Bad media thumbnail key"	404 (not found)
1146	"Slideshow end" – reached the end of the slideshow	404 (not found)
1147	"Media busy" – another media item is already playing	409 (conflict)
1148	"Not a slideshow" – the request is only valid for slideshow items	400 (bad request)
1149	"Already playing" – you tried to play an item that is already playing	409 (conflict)
1150	"Already stopped" – you tried to stop a media item that was not playing	409 (conflict)
1151	"API is not enabled" – the relevant API is not enabled	401 (unauthorised)
1152	"Could not set song number" – it was not possible to set the song number (perhaps it is already playing or the number is not a valid song number)	409 (conflict)
1153	"Could not identify song number" – the song number specified in the URI is not a number	400 (bad request)
1154	"Invalid background music command" – the background music command is unknown or not applicable in the current context	409 (conflict)

1155	"Subscription address not found" – a bad address was specified in an event notification subscription request	400 (bad request)
1156	"Subscription port not found" – a bad port number was specified in an event notification subscription request	400 (bad request)
5128	"Unknown error" – any other errors	500 (internal server error)

Note that all errors are logged in the Windows event log.

3. UDP Broadcast Response

You can automatically retrieve the SoundBox server IP address by sending a UDP network broadcast to port 8095 with the ASCII-encoded text "SoundBox" in the payload. SoundBox responds to such a broadcast with a simple ASCII string containing the text "SoundBox" followed by a single tab character and the IP address of the host machine, e.g.:

```
SoundBox    192.168.1.94
```

This is a useful alternative (or addition) to iteratively probing the network or requiring the client to enter the IP address in a device.

You must enable this feature in the SoundBox Options, Settings, Miscellaneous section.

Note that UDP is not a reliable protocol and does not guard against corruption of network packets, nor guarantee the delivery of packets. You can perform your own error checking by broadcasting several times and comparing the responses – if they all give the same result then the value is very likely correct.

4. Event Notifications

The APIs described above allow you to request data from SoundBox or initiate commands. The event notifications mechanism works the other way around; SoundBox pushes notifications to your application when specific events occur, such as when a video is started, when an image is shown, etc. Notifications can be used to help integrate SoundBox with other audio-visual tools such as HDMI switches, cameras, video production software, etc. For example, it is possible to create an application that listens for SoundBox events and switches a video stream source to the PC's media display when an image or video is shown.

The notifications are delivered using JSON over TCP, and all have a similar content as illustrated below:

```
{
  "event": "StartVideo",
  "stamp": "2016-10-28T11:38:42",
  "server": "192.168.0.12"
}
```

The elements of the Event object are described below:

- **event** – The SoundBox event
- **stamp** – the date and time at which the notification was sent (in ISO 8601 format)
- **server** – the IP address of the SoundBox machine

Events include the following (most are self-explanatory):

- **StartSong** – start of song
- **StopSong** – end of song
- **StartVideo** – start of video
- **StopVideo** – end of video
- **PauseVideo** – video paused
- **RestartVideo** – video restarted after a pause
- **OpenBrowser** – browser opened to display a website or PDF
- **CloseBrowser** – browser closed
- **StartAudio** – audio file started
- **StopAudio** – audio file stopped
- **ShowImage** – image shown (but not applicable to background image)
- **HideImage** – image hidden
- **StartSlideShow** – slideshow started
- **StopSlideShow** – slideshow stopped
- **PauseSlideShow** – slideshow paused
- **RestartSlideShow** – slideshow restarted after a pause
- **ShowLyrics** – lyrics displayed
- **HideLyrics** – lyrics hidden
- **StartCountdown** – meeting countdown started
- **StopCountdown** – meeting countdown stopped
- **StartBackgroundMusic** – background music started
- **StopBackgroundMusic** – background music stopped
- **CloseApp** – SoundBox closing
- **StartTimer** – a talk timer started
- **StopTimer** – a talk timer stopped
- **StartRecording** – audio recording started
- **StopRecording** – audio recording stopped
- **StartSubscription** – your subscription was successful
- **DropSubscription** – your subscription was dropped because there are too many

Note that there is no supplemental information delivered with each notification; you can use the relevant API to retrieve extra data if required.

4.1 Subscribing

To subscribe to event notifications, send a POST request to the following URL:

```
http://soundbox_machine:8095/api/v7/events/subscribe
```

Include in the body of your request, the TCP port and IP address that your application is listening on. A typical body is shown below:

```
{
  "address": "192.168.0.14",
  "port": 9550
}
```

If you omit the address, SoundBox will try to extract it from the request data.

SoundBox sends a status code, and a JSON representation of the Event object in the body of the response. Once you have subscribed, SoundBox will send event notifications to your application using the address of the subscribing machine and the port you specified in the subscription request. The first event you are likely to receive is a "StartSubscription" event as additional confirmation that the subscription is established. Notifications are dispatched using TCP.

Notes

- You can't subscribe to a subset of the notifications; it's all or nothing!
- SoundBox supports a maximum of 25 subscriptions. If there are 25 subscriptions and you issue another subscription request, SoundBox drops the oldest subscription. If your subscription is dropped, you will receive a "DropSubscription" notification beforehand.
- It is ok to subscribe while already subscribed; SoundBox simply updates your subscription date with the current date and time.

4.2 Unsubscribing

To unsubscribe to event notifications, send a POST request to the following URL:

```
http://soundbox_machine:8095/api/v7/events/unsubscribe
```

Include in the body of your request, your application's endpoint details. A typical body is shown below:

```
{
  "address": "192.168.0.15",
  "port": 9550
}
```

SoundBox sends a status code, and a JSON representation of the Event object in the body of the response.

4.2 Recent Events

You can retrieve a list of the most recent 20 events by sending a GET request to the following URL:

```
http://soundbox_machine:8095/api/v7/events/subscribe
```

You do *not* need to subscribe to event notifications to use this API. The response is a collection of Event objects as illustrated below:

```
[
  {
    "event": "StartSong",
    "stamp": "2016-10-28T13:00:27",
    "server": "192.168.1.94"
  },
  {
    "event": "ShowLyrics",
    "stamp": "2016-10-28T13:00:30",
    "server": "192.168.1.94"
  },
  {
    "event": "HideLyrics",
    "stamp": "2016-10-28T13:03:54",
    "server": "192.168.1.94"
  },
  {
    "event": "StopSong",
    "stamp": "2016-10-28T13:03:54",
    "server": "192.168.1.94"
  },
  {
    "event": "StartTimer",
    "stamp": "2016-10-28T13:04:38",
    "server": "192.168.1.94"
  }
]
```


Note that the *StartSubscription* and *DropSubscription* events are not included in the list of recent events because they are client-specific.

5. Usage Tips

5.1 Network

Try to keep the number of network calls to a minimum. In particular, if you want to dynamically update a timer's value don't attempt to poll the SoundBox server every 100 ms; poll it once to get a timer's value then maintain the display using a client-based timer, perhaps polling again every 2 or 3 seconds just to check that the timer hasn't been stopped (e.g. by the SoundBox operator). You can also make use of the event notifications to reduce polling.

5.2 Starting a Timer

When starting a timer always retrieve a fresh copy of the timer data and check that it is enabled and not already running, and if you are displaying the timer, use the fresh value of the "actualAllocationSecs" property to use as the initial duration of your timer display.

5.3 Synchronising a Client Timer

If you want to display a dynamic timer in your client, there are a few important notes:

1. When SoundBox starts a timer, it delays the *actual* start of countdown by 1500 ms. This is done for the sake of visual clarity when using the external timer monitor – allowing the initial timer value

to fade in before it starts to count down. Accordingly, you should add the 1500 ms delay into your client-side timer.

2. Network latency can make it difficult to keep it exactly in step with the SoundBox timer (by the time you have retrieved the value of the SoundBox timer via a web API call the value may already be 100s of milliseconds old). To do this successfully you can time the API call in your client and deduce the offset required to synchronise. The formula is as follows:

- `offset = ((t1 - t0) + (t2 - t3)) / 2`

where

- ***t0*** = client elapsed timer at request transmission
- ***t1*** = SoundBox elapsed timer at receipt of request
- ***t2*** = SoundBox elapsed timer at response transmission
- ***t3*** = client elapsed timer at receipt of response

In SoundBox, and for the purpose of this calculation, we can safely assume that the ***t1*** and ***t2*** are identical, so the formula becomes:

```
offset = ((t1 - t0) + (t1 - t3)) / 2
```

```
offset = (2t1 - t0 - t3) / 2
```

In your client code you know:

- ***t0*** = the elapsed timer value just before you send the request (=0 if you are starting a timer)
- ***t1*** = the SoundBox elapsed time as returned by the API call
- ***t3*** = the value of ***t0*** plus the duration of the roundtrip API call (which you can time)

3. When a timer reaches zero, SoundBox inserts a 'grace second' before starting to count again. This means that the timer rests at zero for 2 seconds!

5.4 SoundBox Controls

If you are using a client application to control timer and/or song functions, it is still possible for the same functions to be invoked within the SoundBox application. This means that the state of a timer or song control may change and any time. Consider implementing a regular check (e.g. once every 3 seconds) to get the latest relevant information. Ideally, if a timer is started in the SoundBox application, the client should detect this and display the updated information.

Note that the *timerInfo* object includes a *runningIndex* property (from API v2) which can be helpful in keeping the client timer in sync with SoundBox.

5.5 Applications

Several API-enabled SoundBox applications are available and can be a useful resource when developing against the API. Please see the SoundBox website for further details:

http://cv8.org.uk/soundbox