Bose SoundTouch Webservices API

Bose Corporation

Version 1.1.0

Contents

1 Document Version History

Version	*Release Date*	*Description of Changes*
1.0.0	December 5, 2014	• Initial Release
1.0.1	December 17, 2014	• Section 3 updated with a link to the License Agreement • Updated incorrect variable names to remove errant "\" in sections: 6.10, 6.11, 6.14–6.17, 6.20, 7.5 • Corrected WebSockets port to 8080 (previous version incorrectly listed 8090) • Title/description corrections for section 7.2
1.1.0	February 5, 2016	• Minor Corrections to sections 6.3 and 6.4 • Fix various misspellings and typos. • Added clarity around the /select command to show how it can be used to select AUX and Bluetooth sources where available • Added instruction around initiating a WebSockets connection with a speaker • Rearranged the order of the documentation to form more relevant groupings

2 Acronyms and Definitions

Acronyms	*Expanded Term*	*Definition*
API	Application Programming Interface	A definition for how to interact with and use a software component
REST	Representational State Transfer	A common type of web service API that is modeled around resources
WS API	Webservices API	An API made available by a web server
SSDP	Simple Services Discovery Protocol	A discovery protocol that uses unicast and multicast over UDP
MDNS	Multicast Domain Name System	A type of discovery protocol that requires zero configuration
	Bonjour	Apple's implementation of MDNS

3 Contact Info/Legal

For any questions, comments, or suggestions for improvements please email us at SoundTouchAPI@bose.com

Use of this API material is subject to the API License Agreement, which can be found at developers.bose.com/SoundTouch-API-License

4 Overview

These commands are the primary interface to command and control a Bose SoundTouch. They are sent over HTTP on port 8090 to the SoundTouch device you would like to connect to using the GET and POST methods.

4.1 Special types used by the SoundTouch WSAPI

```
ART_STATUS {
    INVALID
    SHOW_DEFAULT_IMAGE
    DOWNLOADING
    IMAGE_PRESENT
}

BOOL: "true" or "false"

INT: a 32-bit integer

IPADDR: an IP address, represented as a string

KEY_VALUE {
    PLAY
    PAUSE
    STOP
    PREV_TRACK
    NEXT_TRACK
    THUMBS_UP
    THUMBS_DOWN
    BOOKMARK
    POWER
    MUTE
    VOLUME_UP
    VOLUME_DOWN
    PRESET_1
    PRESET_2
    PRESET_3
    PRESET_4
    PRESET_5
    PRESET_6
    AUX_INPUT
    SHUFFLE_OFF
    SHUFFLE_ON
    REPEAT_OFF
    REPEAT_ONE
    REPEAT_ALL
    PLAY_PAUSE
    ADD_FAVORITE
    REMOVE_FAVORITE
    INVALID_KEY
}

KEY_STATE {
  press
  release
}

MACADDR: a MAC address, upcased, represented as a string

PLAY_STATUS {
    PLAY_STATE
    PAUSE_STATE
    STOP_STATE
    BUFFERING_STATE
```

```
    INVALID_PLAY_STATUS
}

PRESET_ID: An integer, 1 through 6 inclusive

SOURCE_STATUS {
    UNAVAILABLE
    READY
}

STRING: any valid XML-escaped string

UINT: a 32-bit unsigned integer

UINT64: a 64-bit unsigned integer

URL: a URL, encoded as a string

Any get* command results in a HTTP GET command
Any set* command results in a HTTP POST command, i.e. requires a payload
```

5 General Status and Errors

For calls that do not have a special return payload, the default response is:

```
<status>$STRING</status>
```

For calls that can produce errors, the error response is:

```
<errors deviceID="$STRING">
    <error value="$INT" name="$STRING" severity="$STRING">$STRING</error>
    ...
</errors>
```

For malformed requests, i.e., wrong value the response is:

```
<error>XML parse error (1:116): Error reading Attributes.</error>
```

```
<errors deviceID="D05FB8A9591D"><error value="1019" name="CLIENT_XML_ERROR"
    severity="Unknown">1019</error></errors>
```

6 API Methods/URLs

6.1 /key

Description: Keys are used as a simple means to interact with the SoundTouch speaker. For a full listing of supported keys please see the list under KEY_VALUE in section 4.1

Send a remote button press to the device

GET:
N/A

POST:

```
<key state="$KEY_STATE" sender="$KEY_SENDER">$KEY_VALUE</key>
```

In general, it is good practice to send 2 discrete HTTP POST calls, the first using "press" as the key state, and the second using "release" as the key state. Doing so simulates both the press and release action of clicking a key. Possible values for "$KEY_STATE" are "press" or "release".

The back to back message bodies will look like the following:

```
<key state="press" sender="Gabbo">$KEY_VALUE</key>
```

```
<key state="release" sender="Gabbo">$KEY_VALUE</key>
```

6.2 /select

Description:

Use this /select API to select AUX or Bluetooth sources when available. Sources available via this /select API will vary based on product. Use the /sources API to view the availability for the device.

GET:

N/A

POST:

Examples:

Sources available via this /select API will vary based on product.

Use the /sources API to view the availability for the device.

Below are some samples for Bluetooth and AUX

```
<ContentItem source="AUX" sourceAccount="AUX"></ContentItem>
```

```
<ContentItem source="AUX" sourceAccount="AUX3"></ContentItem>
```

```
<ContentItem source="BLUETOOTH"></ContentItem>
```

6.3 /sources

Description:

List all available content sources

GET:

```
<sources deviceID="$MACADDR">
    <sourceItem source="$SOURCE" sourceAccount="$STRING"
        status="$SOURCE_STATUS">$STRING</sourceItem>
    ...
</sources>
```

POST:
N/A

6.4 /bassCapabilities

Description: Some speakers do not support the ability to customize the bass levels, use this to find out whether bass customization is supported

Get or set bassCapabilities

GET:

```
<bassCapabilities deviceID="$MACADDR">
   <bassAvailable>$BOOL</bassAvailable>
   <bassMin>$INT</bassMin>
   <bassMax>$INT</bassMax>
   <bassDefault>$INT</bassDefault>
</bassCapabilties>
```

POST:
N/A

6.5 /bass

Description: Sets or gets the current bass setting for a particular speaker. This may or may not be a supported capability, use the /bassCapabilities to find out whether a speaker supports bass configuration

Get or set bass

GET:

```
<bass deviceID="$MACADDR">
    <targetbass>$INT</targetbass>
    <actualbass>$INT</actualbass>
</bass>
```

POST:

```
<bass>$INT</bass>
```

6.6 /getZone

Description:

Gets the current state of the multi-room zone from particular device

GET:

```
<zone master="$MACADDR">
    <member ipaddress="$MASTER_IPADDR">"$MASTER_MACADDR"</member>
    <member ipaddress="$SLAVE1_IPADDR">"$SLAVE1_MACADDR"</member>
    ...
</zone>
```

6.7 /setZone

Description:

Creates a multi-room zone

GET:
N/A
POST:

```
<zone master="$MACADDR" senderIPAddress="$IPADDR">
    <member ipaddress="$IPADDR">$MACADDR</member>
    ...
</zone>
```

6.8 /addZoneSlave

Description:

Add a slave to a “play everywhere” zone

GET:
N/A
POST:

```
<zone master="$MACADDR">
    <member ipaddress="$IPADDR">$MACADDR</member>
    ...
</zone>
```

6.9 /removeZoneSlave

Description:

Take a slave out of a “play everywhere” zone

GET:
N/A
POST:

```
<zone master="$MACADDR">
    <member ipaddress="$IPADDR">$MACADDR</member>
    ...
</zone>
```

6.10 /now_playing

Description:

Gets all info about the currently playing media

GET: