Custom Integration API

The information contained in this manual is the property of Lenbrook Industries. It is to be solely intended for professional use. Lenbrook Industries assumes no responsibility for the accuracy of the protocol. The protocol is provided “as-is”, with all faults and without warranty of any kind, either expressed or implied.

API Use Policy

By accessing the APIs, you agree to this API Use Policy (the "Policy") and our Terms. We provide these APIs to allow companies and people to build on and benefit from our Service by creating software, services, or modules that connect to our platform or have access to the data within our platform via our APIs (an "Integration"). This Policy is and will be treated as part of our Terms.

The software is provided "AS IS", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages, or other liability, whether in an action of contract, Tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

Permitted Use

You may not use the API to send spam or take any actions that violate our Acceptable Use Policy and our Standard Terms of Use. You will comply with all applicable laws (including privacy laws and United States export control laws, European GPR and regulations and economic sanctions laws and regulations). You will follow all documentation we provide for the APIs. You will not attempt to hack or change the way the Service functions. We may monitor your use of the APIs for compliance with these rules, and we may deny you access to the API if you violate this Policy.

Privacy

Your Integration must display a privacy policy for users detailing the information you will collect from them when they use the Integration. You will only access a user's data to the extent permitted by the user and explained in your privacy policy. You must immediately delete a user's data if the user requests deletion or terminates their account with you.

Security

You will implement and maintain appropriate technical and organizational security measures to protect and to preserve the security, integrity, and confidentiality of the data. These security measures shall prevent the unauthorized access or disclosure of personal or confidential data that you process.

Ownership

We own all rights, titles, and interest in the Service and the APIs, including all intellectual property rights, marks, code, and features. You will not infringe, reverse engineer, or copy our code, design, or content. You will not access our APIs to compete with our Service. Any rights not expressly granted by this Policy are withheld, so if you do not see it here, then it is not a right we are allowing you.

Use of Marks

You may not use our name and marks (meaning our logos, brands, and copyrighted images) in any way. You may not alter or remove any proprietary notices in our marks. You will not use our name or marks in your Integration name or logo, or in any way that implies an endorsement by us.

Practical Use of Marks

These guidelines explain how you must practically use our name, marks, and brand assets at all times. Your use indicates your acceptance of these guidelines, and you understand that your use in violation of these guidelines will result in automatic termination of your permission to use our name, marks, and brand assets.

- Use of our name, marks and brand assets must be expressly authorized in writing.
- Do not change, modify, distort, copy, or imitate our Brand Assets in any way, including changing the colour, rotating and/or stretching. In other words, our Brand Assets must be kept in their original forms.
- Do not give our name, marks and brand assets undue prominence compared to your name and logo.
- Do not display our name, marks, and brand assets next to, or in any form of competitive marketing, without our express consent.
- Your use must not mislead consumers as to our sponsorship of, affiliation with or endorsement of your company or your products or services.
- Our name, marks and brand assets are our exclusive property. All goodwill that results from your use will be solely to our benefit. You will not take any action that is at odds with our rights or ownership.
- Our name, marks and brand assets must be used in a respectful manner and may not be used in a way that harms us, our products, or services, or in a manner which, in our opinion, lessens or otherwise damages our reputation or the goodwill in our name, marks and brand assets. In other words, please do not associate our assets with any illicit or illegal activities or use them in a way that is deceptive or harmful.

Examples of acceptable use:

“[YOUR PRODUCT NAME] (compatible with / works with BluOS)”

Examples of unacceptable use

“[YOUR PRODUCT NAME] - BluOS”

“BluOS – [YOUR PRODUCT NAME]”

“[YOUR PRODUCT NAME] – Powered with BluOS”

Marketing and Press Releases

After your application is approved, it may be listed on our web properties. We generally will not co-publish press releases or contribute to co-marketing of your application.

Before you distribute a press release about your app, make sure that you reach out to us at [EMAIL]. If you will be mentioning BluOS, then we will need to review the release. We recommend reaching out with your final press release as soon as possible.

Disclaimer

To the maximum extent permitted by law, we provide the APIs as-is. That means we do not provide warranties of any kind, either express or implied, including but not limited to merchantability and fitness for a particular purpose.

Updates

We may update or modify the APIs and this Policy from time to time by posting the changes on this site or notifying you via email. These changes may affect your use of the APIs or the way your Integration interacts with the API. If we make a change that is unacceptable to you, you should stop using the APIs.

Confidentiality

You may have access to confidential, proprietary, and non-public information specific to the APIs ("Confidential Information"). You may use this information only to build with the APIs. You will not disclose the Confidential Information to anyone without our written consent, and you will protect the Confidential Information from unauthorized use and disclosure in the same way you would protect your own confidential information.

Indemnification

You will indemnify and hold us and our Team harmless from any losses (including attorney fees) that result from third-party claims that relate to your use of the API.

The Rest

This Policy does not create or imply any partnership, agency, or joint venture. This Policy will apply for as long as you use the APIs or until terminated in accordance with our Terms. In the event of a conflict between this Policy and the Standard Terms of Use, the Standard Terms of Use shall control.

Revision History		
Version	**Date**	**Description**
1.0	6/17/2019	First release
1.2	01/12/2022	Added soft reboot, doorbell chime, volume up/down, move track in queue, and direct input commands. Added note to Appendix LSDP.
1.4	04/26/2022	Added mute command; modified direct input commands for HUB; updated Play command to play streamed custom audio.
1.5	07/18/2022	Added Bluetooth command; Updated LSDP to add class 5 to 8; Added “Practical Use of Marks” in API Use Policy.

Table of Contents

1. Introduction

BluOS™ is an advanced operating system and music management software that allows you to access and stream lossless music up to 24-bit/192kHz to every room using your home network. BluOS can be found in products from Bluesound, NAD Electronics, DALI Loudspeakers and others.

This document was created to assist developers and system integrators who are working in the custom integration (CI) marketplace. It contains a subset of the API requests documented in the full BluOS API Control Protocol.

All the requests described in this document are sent as HTTP GET requests. The parameters are a standard URL encoded name/value pair. BluOS players receive these commands and then respond with UTF-8 encoded XML data.

All requests are in the form of http://<player_ip>:<port>/<request> where:

- player_ip is the IP address of the BluOS player (e.g., 192.168.1.100)
- port is the TC port used for communications. Port 11000 is used for all BluOS players, with the exception of the CI580. The CI580 has four streamers nodes in one chassis, where node 1 uses port 11000, node 2 uses port 11010, node 3 uses port 11020, and node 4 uses port 11030. The actual port to use should be discovered by use of the MDNS protocol using the services musc.tcp and musp.tcp.
- request is the actual BluOS command or query (e.g., Play)

Note: This document will use http://192.168.1.100:11000 as player IP and port in all examples.

2. Status Queries

Status queries are used to query a BluOS player.

BluOS provides two mechanisms for doing status queries; regular polling and long polling. Regular polling returns the query result immediately. Long polling keeps a connection for a specified time, and only returns a query result when information has changed. Long polling can greatly reduce the number of calls to a player.

When long-polling is not being used then clients should restrict their polling rate to at most one request every 30 seconds. When long-polling is being used then a client must not make two consecutive requests for the same resource less than one second apart, even if the first request returns in less than one second.

Long polling requests take two parameters: timeout and etag. timeout is the duration of the long-poll request and etag is taken from the previous response (an attribute in the root element of the response).

In general it is only necessary to have a long-poll active for one of /Status or /SyncStatus. The /Status response include an element (<syncStat>) that indicates whether /SyncStatus has changed. /SyncStatus should be polled if only the name, volume and grouping status of a player is of interest. /Status should be polled if current playback status is needed.

2.1 Playback Status

Description

The /Status endpoint queries volume and playback information. This query returns many response attributes, some of which are not applicable to this document. Undocumented responses should be ignored.

Request

/Status?timeout=*seconds*&etag=*etag-value*

Parameters	Description
timeout	Optional parameter used with long polling. Recommended polling interval is 100 seconds and should be limited to a rate of 60 seconds or so and never faster than 10 seconds.
etag	Optional parameter used with long polling. This is the etag attribute from the previous /Status call response.

Response

```
<status etag="4e266c9fbfba6d13d1a4d6ff4bd2e1e6">
```

<album>÷ (Deluxe)</album>

<artist>Ed Sheeran</artist>

<canMovePlayback>true</canMovePlayback>

<canSeek>1</canSeek>

<cursor>159</cursor>

<fn>Deezer:142986206</fn>

<image>/Artwork?service=Deezer&songid=Deezer%3A142986206</image>

<indexing>0</indexing>

<mid>187</mid>

<mode>1</mode>

<name>Perfect</name>

<pid>1054</pid>

<prid>0</prid>

<quality>320000</quality>

<repeat>2</repeat>

<service>Deezer</service>

<serviceIcon>/Sources/images/DeezerIcon.png</serviceIcon>

<shuffle>0</shuffle>

<sid>8</sid>

<sleep/>

<song>19</song>

<state>pause</state>

<streamFormat>MP3 320 kb/s</streamFormat>

<syncStat>5</syncStat>

<title1>Perfect</title1>

<title2>Ed Sheeran</title2>

<title3>÷ (Deluxe)</title3>

<totlen>263</totlen>

<volume>4</volume>

<secs>35</secs>

</status>

NOTE: not all response attributes are listed in the following chart. Other elements may be present and should be ignored.

Response Attributes	Description
etag	Attribute of the response root element. Opaque value used with long-polling to detect response changes. If the value has not changed since the previous response then the response is guaranteed not to have changed (but see also secs below)
alarmsecondsremaining	If playback is the result of an alarm, then this is how many seconds before it will stop.
action	See Actions for Streaming Radio Stations section for description.
album	Album name of the current active track. Also see title1 attribute.
artist	Artist name of the current active track. Also see title1 attribute.
battery	Shown if the player has a battery pack. Includes attributes: • level – state of charge, percent • charging – 1 if currently charging • icon – URL of player image indicating current charge state
canMovePlayback	Is true if it is possible to move the current playing or paused content to another player.
canSeek	If 1 then it is possible to scrub through the current track, in the range 0..totlen, by use of the seek parameter to /Play. For example: /Play?seek=34.
db	Volume level in dB.
groupName	Name of the group. The player must be the primary player in the group.
groupVolume	Volume level of the group. The player must be the primary player in the group.
image	URL of image associated with the current audio (album, station, input, etc.)
mute	Mute state. Set to 1 if volume is muted.
muteDb	if the player is muted, then this contains the unmuted volume in dB.
muteVolume	If the player is muted, then this contains the unmuted volume level. Values are from 0 to 100.
name	The title of the current playing audio track. Also see title1 attribute.

Response Attributes	Description
notifyurl	URL for a pop up notification.
pid	The unique play queue id. It matches the id attribute of the /Playlist response. If the play queue is changed this number will change.
prid	The unique preset id. It matches the prid attribute in the /Presets response. If a preset is changed this number will change indicating that any cached response to /Presets should be purged.
quality	Quality of the playing source audio: • cd - losless audio at CD quality • hd – lossless audio with higher resolution that CD quality or sample-rate of 88200 samples/s or more • dolbyAudio – DolbyDigital or AC3 • mqa – valid MQA audio decoded • mqaAuthored - valid MQA-Authored audio decoded A numeric value is the approximate bitrate of a compressed audio source quality of the file.
repeat	0, 1, or 2. 0 means repeat play queue, 1 means repeat a track, and 2 means repeat off
secs	Number of seconds the current audio track has been played. This value is not used in the calculation of etag and progress will not of itself cause a return from a long-polling call. Clients are required to increment the playback position, when state is play or stream, based on the interval since the response.
service	The service id of the current audio. This is not a value for displaying in a UI, as the actual string may differ from the official service name.
serviceIcon	URL of the current service icon.
shuffle	0 or 1. 0 means shuffle off and 1 means shuffle on.
sleep	The minutes remaining before the sleep timer activates.
song	The position of the current track in the play queue. Also see streamUrl.
state	The current player state. It could be play, pause, stop, stream, connecting, etc. /Play can be used to resume when in a pause state but not when in stop state.

Response Attributes	Description
	play and stream should be considered to have the same meaning. Also see streamUrl.
stationImage	URL for the image of a radio station, if the current audio is a radio station, e.g., Deezer radio. It may be the same as image.
streamFormat	Format of the audio.
streamUrl	The presence of this element should be treated as a flag and its contents as an opaque value. If present it indicates: • the play queue is not the source of the current audio (song is irrelevant) • shuffle and repeat are not relevant and should be removed from any UI if possible • next and previous are not available (but see also actions)
syncStat	The unique id to indicate any change in /SyncStatus response. It matches the syncStat attribute of the /SyncStatus response. It changes whenever there is a change in Sync Status.
title1	The first line of information describing the current audio. title1, title2 and title3 MUST be used as the text of any UI that displays three lines of now-playing metadata. Do not use values such as album, artist and name.
title2	The second line of information describing the current audio.
title3	The third line of information describing the current audio.
totlen	Total length of the current track, in seconds.
twoline_title1	The first of two lines describing the current audio. twoline_title1 & twoline_title2, if present, MUST be used as the text of any UI that displays two lines of now-playing metadata.
twoline_title2	The second of two lines describing the current audio.
volume	The player volume level in percentage; -1 means player volume fixed.
secs	The number of seconds the current audio track has been played.

Example

http://192.168.1.100:11000/Status

Gets the playback status of the player.

http://192.168.1.100:11000/Status?timeout=100&etag=4e266c9fbfba6d13d1a4d6ff4bd2e1e6

Gets the playback status of the player using long-polling. A result is only returned before the timeout of 100 seconds if the player's status has changed. Otherwise, the result is returned after 100 seconds.

2.2 Player and Group Sync Status

Description

The SyncStatus query returns player information and player grouping information. This query returns many response attributes, some of which are not applicable to this document. Undocumented responses should be ignored.

Request

/SyncStatus?timeout=*seconds*&etag=*etag-value*

Parameters	Description
timeout	Optional parameter used with long polling. It is the polling interval in seconds. Recommended polling interval is 180 seconds.
etag	Optional parameter used with long polling. This is the etag attribute from the previous /SyncStatus call response.

Response

<SyncStatus icon="/images/players/P300_nt.png" volume="4" modelName="PULSE" name="PULSE-0278" model="P300" brand="Bluesound" etag="23" outlevel="-62.9" schemaVersion="25" initialized="true" group="PULSE-0278 + 2" syncStat="23" id="192.168.1.100:11000" mac="90:56:82:9F:02:78">

<master port="11000">192.168.1.100</master>

<slave port="11000" id="192.168.1.153"/>

<slave port="11000" id="192.168.1.234"/>

…..

</SyncStatus>

NOTE: not all response attributes are listed in the following chart. Other elements may be present and should be ignored.

Response Attributes	Description
battery	Shown if the player has a battery pack. Includes attributes: • level – state of charge, percent

Response Attributes	Description
	• charging – 1 if currently charging • icon – URL of player image indicating current charge state
brand	Player brand name.
db	Volume level in dB.
etag	Tag of the /SyncStatus response, used for long polling.
group	The group name.
icon	URL that contains the player icon image.
id	Player IP and port.
initialized	True means the player is already setup, false means the player needs setup. The player must be setup with the BluOS Controller app.
mac	Player unique id for network interface. May be a MAC address.
master	Master player IP address. Only present if a player is a secondary player in a group. Attributes: • port - port number. • reconnecting – true if trying to reconnect to primary player
model	Player model id.
modelName	Player model name.
mute	Set to 1 if volume is muted.
muteDb	If the player is muted, then this is the unmuted volume level in dB.
muteVolume	If the player is muted, then this is the unmuted volume level (0..100).
name	Player name.
schemaVersion	Software schema version.
slave	Secondary player(s) IP addresses. Only present if player is the primary player of a group. There can be multiple secondary players. Attributes: • id – IP address • port – port number

Response Attributes	Description
syncStat	id of sync status. It's changed whenever any item in /SyncStatus response is changed. Match with <syncStat> element in /Status response.
volume	Volume level on a 0..100 scale. -1 means fixed volume.
zone	Name of fixed group.
zoneMaster	If the player is the primary player in a fixed group, this is set to true.
zoneSlave	If the player is a secondary player in a fixed group, this is set to true.

Example

http://192.168.1.100:11000/SyncStatus

Gets the player and group status of the player.

http://192.168.1.100:11000/SyncStatus?timeout=100&etag=4e266c9fbfba6d13d1a4d6ff4bd2e1e6

Gets the player and group status of the player using long-polling. A result is only returned before the timeout of 100 seconds if the player's status has changed. Otherwise, the result is returned at 100 seconds.

3. Volume Control

Adjusts the volume level of a player. Also used to mute a player.

3.1 Set Volume

Description

This request queries or sets the player volume.

All command variants, whether using 0..100 level, absolute dB or relative dB parameters, are constrained to values that result in a level within the configured available volume range, which is typically -80..0. The volume range can be adjusted using the BluOS Controller app, on the Settings -> Player -> Audio page.

The query supports long polling (not illustrated below).

Request

/Volume

/Volume?level=*level*&tell_slaves=*on_off*

/Volume?mute=*on_off*&tell_slaves=*on_off*

/Volume?abs_db=*db*&tell_slaves=*on_off*

/Volume?db=*delta-db*&tell_slaves=*on_off*

Parameters	Description
level	Set the absolute volume level of the player. It is an integer from 0 -100.
tell_slaves	Applies to grouped players. If set to 0, only the currently selected player changes volume. If set to 1, then all players in the group change volume.
mute	If set to 0, the player is muted. If set to 1, the player is unmuted.
abs_db	Set the volume using a dB scale.
db	Do a relative volume change using a dB volume scale. db can be a positive or negative number.

Response

<volume db="-49.9" mute="0" offsetDb="0" etag="6213593a6132887e23fe0476b9ab2cba">15</volume>

Response Attributes	Description

Response Attributes	Description
db	Volume level in dB.
mute	1 if the player is muted, 0 if the player is unmuted.
muteDb	If the player is muted, then this is the unmuted volume level in dB.
muteVolume	If the player is muted, then this is the unmuted volume level (0..100).
volume	The current volume level: 0..100 or -1 for fixed volume.

Example

http://192.168.1.100:11000/Volume?level=15

Sets the player volume level to 15 (out of 100).

http://192.168.1.100:11000/Volume? tell_slaves=1&db=2

Increases the volume of master player 192.168.1.100, and all of the secondary players in that group, by 2 dB.

http://192.168.1.100:11000/Volume?mute=1

Mutes the player.

3.2 Volume Up

Description

This request increases volume by certain dB (typical value is 2dB).

Request

/Volume?db=db_value

Parameters	Description
db	Volume increase steps in dB (typical value 2dB)

Response

<volume db="-25" mute="0" offsetDb="6" etag="a071a168fac1c879b1de291720c8a4b8">27</volume>

Response Attributes	Description
db	Volume level in dB.
mute	1 if the player is muted, 0 if the player is not muted

Response Attributes	Description
offsetDb	
etag	

Example

http://192.168.1.100:11000/Volume?db=2

Increase the volume by 2dB.

3.3 Volume Down

Description

This request decreases volume by certain dB (typical value is -2dB).

Request

/Volume?db=-db_value

Parameters	Description
db	Volume increase steps in dB (typical value -2dB)

Response

<volume db="-25" mute="0" offsetDb="6" etag="a071a168fac1c879b1de291720c8a4b8">27</volume>

Response Attributes	Description
db	Volume level in dB
mute	1 if the player is muted, 0 if the player is not muted
offsetDb	
etag	

Example

http://192.168.1.100:11000/Volume?db=-2

Decrease the volume by 2dB.

3.4 Mute On

Description

This request sets player to mute.

Request

/Volume?mute=1

Parameters	Description
mute	Set to 1 to mute player

Response

<volume muteDb="-43.1" db="-
100" muteVolume="11" mute="1" offsetDb="0" etag="2105bed56563d9da46942a696cfadd63">0</volume
>

Response Attributes	Description
muteDb	Volume level in dB before mute
db	Volume level in dB
muteVolume	Volume level in percent before mute
mute	1 means the player is muted
offsetDb	
etag	

Example

http://192.168.1.100:11000/Volume?mute=1

3.5 Mute Off

Description

This request sets player to unmute.

Request

/Volume?mute=0

Parameters	Description
mute	Set to 0 to unmute player

Response

<volume db="-43.1" mute="0" offsetDb="0" etag="e72d53db17baa526ebb5ee9c26060b1f">11</volume>

Response Attributes	Description
db	Volume level in dB
mute	0 means the player is not muted
offsetDb	
etag	

Example

http://192.168.1.100:11000/Volume?mute=0

4. Playback Control

These commands are used for basic playback control. Commands include play, pause, stop, skip, back, shuffle, and repeat.

4.1 Play

Description

Start playback of the current audio source. Optional parameters allow a jump into the audio tracks, and an input to be selected, before starting audio playback.

Request

/Play

/Play?seek=*seconds*

/Play?url=encodedS*treamURL*

Parameters	Description
seek	Jump to specified position in the current track. Only valid if /Status response includes <totlen>. Cannot be used with inputType and index parameters.
encodedStreamURL	URL of streamed custom audio. It has to be URL encoded.

Response

<state>play</state>

<state>stream</state>

Response Attributes	Description
state	The state after executing the command. See /Status response state attribute for more details.

Example

http://192.168.1.100:11000/Play

Start audio playback of the current track.

http://192.168.1.100:11000/Play?seek=55

Start audio playback at 55 seconds into the track.

192.168.1.125:11000/Play?url=https%3A%2F%2Fwww%2Esoundhelix%2Ecom%2Fexamples%2Fmp3%2FSoundHelix-Song-1%2Emp3

Start audio playback of an online mp3 audio.

4.2 Pause

Description

Pause the current playing audio.

If an alarm is playing, and it has a timeout, then the alarm timeout is canceled.

Request

/Pause

/Pause?toggle=1

Parameters	Description
toggle	If set to 1, then the current pause state is toggled.

Response

<state>pause</state>

Response Attributess	Description
state	The state after executing the command. See /Status response state attribute for more details.

Example

http://192.168.1.100:11000/Pause

Pauses the currently playing audio.

4.3 Stop

Description

Stop the current playing audio. If an alarm is playing, and it has a timeout, then the alarm timeout is canceled

Request

/Stop

Parameters	Description

Parameters	Description
None	

Response

<state>stop</state>

Response Attributes	Description
state	“stop” means the current audio is stopped.

Example

http://192.168.1.100:11000/Stop

Stops the currently playing audio.

4.4 Skip

Description

Skip to the next audio track in the play queue

When playing from the play queue, it will skip to the next track in the queue. If the current track is the last one in the queue, calling /Skip will go to the first track in the queue. . It will skip to the next or first track in the queue regardless of the state of the repeat setting.

To determine if you are using the play queue, verify that there is no <streamUrl> entry in the /Status response. Then you can use the /Skip command.

You can also skip tracks for some streaming radio stations. These are handled with the /Action command.

Some sources such as TuneIn and Optical Input do not support a skip option. These sources will have a <streamURL> entry but no skip action name in the /Status response.

Request

/Skip

Parameters	Description
None	

Response

<id>21</id>

Response Attributes	Description
id	The track id after executing the skip command. Refer to the /Status response song attribute for details.

Example

http://192.168.1.100:11000/Skip

Skip to the next track.

4.5 Back

Description

If a track is playing and has been playing for more than four seconds then back will return to the start of the track. Otherwise the back command will go to the previous song in the current playlist. If on the first song in the playlist calling back will go to the last song. It will go to the previous or first track in the queue regardless of the state of the repeat setting.

To determine if you are using the play queue, verify that there is no <streamUrl> element in the /Status response. Then you can use the /Back command.

You can also go back tracks for some streaming radio stations. These are handled with the /Action command.

Some sources such as TuneIn and Optical Input do not support a back option. These sources will have a <streamUrl> element but no skip action name in the /Status response.

Request

/Back

Parameters	Description
None	

Response

<id>19</id>

Response Attributes	Description
id	The track id after executing the back command. Refer to the /Status response song attribute for details.

Example

http://192.168.1.100:11000/Back

Go back to the beginning of the track, or to the previous track.

4.6 Shuffle

Description

The shuffle command creates a new queue by shuffling the current queue. The original (not shuffled) queue is retained for restore when shuffle is disabled.

Request

/Shuffle?state=0|1

Parameters	Description
state	• 0 to disable shuffle • 1 to enable shuffle. Has no effect if queue is already in shuffled state. See /Status response <shuffle> element.

Response

<playlistname="Calm Piano"modified="0"length="160"shuffle="1"id="1051"/>

Response Attributes	Description
modified	1 means the queue has been modified since being loaded. 0 means not.
length	Total number of tracks in the current queue.
shuffle	The shuffle state. 1 means the current queue is shuffled. 0 means the current queue is not shuffled.
id	The current queue id. It changes whenever the play queue is modified.

Example

http://192.168.1.100:11000/Shuffle?state=1

Shuffles the current play queue.

4.7 Repeat

Description

Sets repeat options. Repeat has three states; 0 means repeat the current queue, 1 means repeat the current track and 2 means do not repeat. All repeats are indefinite, that is, they do not stop.

Request

/Repeat?state=0|1|2

Parameters	Description
state	• 0 to repeat the entire play queue • 1 to repeat the current track • 2 to turn repeat off

Response

<playlist length="60" id="1764" repeat="1"/>

Response Attributes	Description
length	The total number of tracks in the current play queue.
id	The current queue id. It changes whenever the play queue is modified.
repeat	The current repeat state.

Example

http://192.168.1.100:11000/Repeat?state=1

Repeats the current playing track.

4.8 Actions for Streaming Radio Stations

Description

Actions allow you to skip forward, go back, love and ban tracks on select streaming radio stations, such as Slacker or Radio Paradise or Amazon Music Prime Stations. Streaming radio stations do not load tracks into the play queue. Instead, they provide a URL that you can use to accomplish the desired function.

Skip will go to the next track. Back will go to the previous track. Love will flag the track as liked within the music service. Ban will skip to the next track and flag the track as disliked within the music service.

If there is a <streamUrl> entry in the /Status response, and an appropriate action, you can do these functions. The action will contain the URL that is used to execute the action.

Here is an example from the /Status response of a player playing Slacker radio:

```
<actions>
    <action name="back"/>
    <action name="skip" url="/Action?service=Slacker&skip=4799148"/>
```

<action icon="/images/loveban/love.png" name="love" notification="Track marked as favorite" state="-1" text="Love" url="/Action?service=Slacker&love=4799148"/>

<action icon="/images/loveban/ban.png" name="ban" notification="Track banned from this station" state="-1" text="Ban" url="/Action?service=Slacker&ban=4799148"/>

</actions>

In this example, back is not available, but skip, love and ban are possible.

Request

/Action?service=*service-name*&action=*action-URL*

Note: The specific request details (endpoint and parameters) are given by the respective <action> element. The commands in the Example section below all use /Action but any URI is possible.

Parameters	Description
	Provided in <action> element.

Response

For the response, you receive an action acknowledgement. For skip and back, you receive:

<skip/>

<back/>

For love you receive:

<love>1</love>

For ban you receive:

<love skip="1">0</love>

Response Attributes	Description
response	If the root element of the response is <response> then the text node is a notification to show to the user. If an alternative root element is returned and the <action> included a notification attribute then that notification should be shown.

Example

http://192.168.1.100:11000/Action?service=Slacker&skip=10965139

Skips to the next Slacker radio track.

http://192.168.1.100:11000/Action?service=Slacker&ban=33332284

Bans the current playing Slacker radio track and skips to the next track.

5. Play Queue Management

One mode of operation of a player is to load tracks into a play queue, and then play the tracks from that play queue. These commands allow you to view and manage the play queue.

5.1 List Tracks

Description

Either return the play queue status, or return information on all tracks in the play queue.

It is not recommended to use this query without either the length or start and end parameters, as otherwise a very long response could be generated.

Request

/Playlist

/Playlist?length=1

/Playlist?start=*first*&end=*last* (retrieve part of the queue, for pagination usually)

Parameters	Description
length=1	Return just the top-level attributes and no track details.
start	First entry in the queue to include in the response, starting from 0.
end	Last entry in the queue to include in the response.

Response

For a play queue status:

```
<playlist>
    <length>13</length>
    <id>243</id>
    <name></name>
    <modified>1</modified>
</playlist>
```

For a play queue listing:

```
<playlist name="Calm Piano" modified="0" length="160" id="1054">
```

<song albumid="61483452" service="Deezer" artistid="6396188" songid="Deezer:487381362" id="25">

<title>2002</title>

<art>Anne-Marie</art>

<alb>2002</alb>

<fn>Deezer:487381362</fn>

</song>

</playlist>

Response Attributes	Description
name	The current play queue name.
modified	0 means the queue hasn't been modified since it was loaded. 1 means the queue has been modified since it was loaded.
length	total number of tracks in the current queue
id	unique id for the current queue state (e.g., 1054). It is same as the <pid> in /Status response.
song	Song is made up of several sub-elements: • albumid = id of the album the track is in • service = music service of the track • artistid = id of the track artist • songid = song id • id = track position in the current queue. If the track is currently selected, track id is same as <song> in /Status response. • title = track name • art = artist name • alb = album name

Example

http://192.168.1.100:11000/Playlist

Lists all the tracks in the play queue.

http://192.168.1.100:11000/Playlist?length=1

Lists only the play queue status.

5.2 Delete A Track

Description

Remove a track from the current play queue.

Request

/Delete?id=*position*

Parameters	Description
id	The track id of the track to be deleted from current play queue.

Response

<deleted>9</deleted>

Response Attributes	Description
deleted	The position in the queue of the track to be removed.

Example

http://192.168.1.100:11000/Delete?id=9

Removes the track at position 9 in the play queue.

5.3 Move A Track

Description

Move a track within the current play queue.

Request

/Move?new=destination&old=origin

Parameters	Description
new	New position on the track being moved.
old	Old position of the track being moved.

Response

<moved>moved</moved>

Response Attributes	Description

Response Attributes	Description
moved	Indicates that the track was moved.

Example

http://192.168.1.100:11000/Move?new=8&old=2

Move the track at position 2 to position 8 in the play queue.

5.4 Clear Queue

Description

Clear all tracks from the current play queue

Request

/Clear

Parameters	Description
None	

Response

<playlist modified="0" length="0" id="1056"/>

Response Attributes	Description
modified	0 means the queue is not modified since loaded, 1 means the queue is modified since loaded.
length	The total number of tracks in the current queue.
id	The unique id for the current queue.

Example

http://192.168.1.100:11000/Clear

This removes all tracks from the play queue.

5.5 Save The Queue

Description

Save the play queue as a named BluOS playlist.

Request

/Save?name=*playlist_name*

Parameters	Description
name	The saved play queue name.

Response

```
<saved>
    <entries>126</entries>
</saved>
```

Response Attributes	Description
entries	The total number of tracks in the saved play queue.

Example

http://192.168.1.100:11000/Save?name=Dinner+Music

This saves the play queue as “Dinner Music”.

6. Presets

Preset requests allow you to list all presets of a player, load a preset, and step up / down presets. Presets must be added and deleted using the BluOS Controller app. Presets can include radio stations, playlists and inputs (e.g. Bluetooth, analog, optical, HDMI ARC).

6.1 List Presets

Description

List all presets on the current BluOS player.

Request

/Presets

Parameters	Description
None	

Response

<presets prid="0">

<preset name="THE HOT 50" url="Load?name=THE HOT 50&service=Deezer&id=707209595" id="4"/>

<preset name="91.1 | JAZZ.FM91 (Jazz)" url="Play?url=TuneIn%3As31229%2Fhttp%3A%2F%2Fopml.radiotime.com%2FTune.ashx%3Fid%3Ds31229%26formats%3Dwma%2Cmp3%2Caac%2Cogg%2Chls%26partnerId%3D8OeGua6y%26serial%3DA4%3A13%3A4E%3A01%3ABD%3A50" id="7"/>

<preset name="Optical Input" url="Play?url=Capture%3Ahw%3A1%2C0%2F1%2F25%2F2" id="16"/>

</presets>

Response Attributes	Description
prid	A unique id of the player presets. It is same as the <prid> in /Status response.
name	The preset name.
id	The preset id.
url	The preset URL. It's the preset source URL used to load the preset.

Example

http://192.168.1.100:11000/Presets

List all the presets on the player.

6.2 Load A Preset

Description

Starts playing a preset. You can select a specific preset number, as well as the next or previous preset. Preset numbers do not have to be sequential, that is, you can have presets 1,2,3 5, 7 and 8. Presets loop around from top to bottom and bottom to top.

Command

/Preset?id=*presetId*|-1|+1

Parameters	Description
id	The id number of the preset to be loaded. The list of available preset id's is found with the Show Presets command. If preset id is +1, it will load the next preset. If preset id is -1, it will load the previous preset.

Response

If the preset is a list of tracks it returns the number of tracks of the preset loaded.

```
<loaded service="Deezer">
    <entries>60</entries>
</loaded>
```

If the preset is a radio it returns the stream state.

```
<state>stream</state>
```

Response Attributes	Description
service	service name of the loaded preset
entries	the number tracks of the preset loaded

Example

http://192.168.1.100:11000/Preset?id=4

Load the preset with preset id 4.

http://192.168.1.100:11000/Preset?id=+1

Load the next preset.

7. Content Browsing and Searching

This section describes commands for music service content browsing and searching.

7.1 Browsing Music Content

Description

Navigate through available music sources, as well as inputs and playlists.

The root element for responses is <browse> unless there is an error response. Most results are a sequence of <item>. In some cases the result is a sequence of <category>, each of which contains a sequence if <item>. All values are provided using attributes. There are no text nodes.

The result of a /Browse call may be an error enclosed in an <error> root element. The detail of the error is provided in one <message> and zero or more <detail> text nodes.

Request

/Browse?key=*key-value*

Parameters	Description
key	Optional parameter. The absence of this parameter will result in a top-level browse. Returns information for levels other than the top level /Browse. Uses the value that is taken from a “browseKey”, “nextKey”, “parentKey”, or “contextMenuKey” attribute value from an earlier response. **Note: key-value must be URL encoded**

Response

Top level browse response:

```
<browse sid="16" type="menu">
    <item image="/images/ci_myplaylists.png" browseKey="playlists" text="Playlists" type="link"/>
    <item image="/images/LibraryIcon.png" browseKey="LocalMusic:" text="Library" type="link"/>
    <item image="/images/InputIcon.png" text="Optical Input"
playURL="/Play?url=Capture%3Ahw%3A1%2C0%2F1%2F25%2F2%2Finput1" inputType="spdif"
type="audio"/>
    <item image="/Sources/images/TuneInIcon.png" browseKey="TuneIn:" text="TuneIn" type="link"/>
    <item image="/Sources/images/SlackerIcon.png" browseKey="Slacker:" text="Slacker" type="link"/>
```

<item image="/Sources/images/TidalIcon.png" browseKey="Tidal:" text="TIDAL" type="link"/>

</browse>

Other level browse response:

<browse sid="16" serviceIcon="/Sources/images/DeezerIcon.png" serviceName="Deezer" service="Deezer" searchKey="Deezer:Search" type="menu">

item browseKey="/Playlists?service=Deezer&genre=0&category=toplist" text="Popular Playlists" type="link"/>

<item browseKey="/Artists?service=Deezer&genre=0&category=toplist" text="Popular Artists" type="link"/>

<item browseKey="/Albums?service=Deezer&genre=0&category=toplist" text="Popular Albums" type="link"/>

item browseKey="/Songs?service=Deezer&genre=0&category=toplist" text="Popular Songs" type="link"/>

</browse>

Response Attributes	Description
Refer to element/attribute tables below	

Example

NOTE: all key parameters must be UTF-8 encoded.

http://192.168.1.100:11000/Browse

This does a top level browse.

http://192.168.1.100:11000/Browse?key=Tidal%3A

Does a second level browse, returning the Tidal categories.

http://192.168.1.100:11000/Browse?key=Tidal%3AmenuGroup%2F3

Does a third level browse, returning the Tidal Masters (Group 3) sub-categories.

http://192.168.1.100:11000/Browse?key=%2FAlbums%3Fservice%3DTidal%26category%3Dmasters

Does a fourth level browse, returning the first set of Tidal Masters albums.

http://192.168.1.100:11000/Browse?key=%2FAlbums%3Fservice%3DTidal%26category%3Dmasters%26start%3D30%26end%3D79

Does another fourth level browse, returning the second set of Tidal Masters albums.

Element	Attribute (and values)		Description
<browse>	serviceIcon		The URI for an icon for the service currently being browsed.
	serviceName		The name of the service currently being browsed, for user display.
	searchKey		A value to use for a key parameter to a /Browse request to search the current service (or some deeper part of the hierarchy). Additionally, the request shall have a q parameter containing the search term.
	nextKey		A value to use for a key parameter to a /Browse request to get the next page of items for the current view. The paging chunk size is not under the control of the API user and no attempt should be made to parse or manipulate the query parameters of this value.
	parentKey		A value to use for a key parameter to a /Browse request to navigate back up the hierarchy if the default back navigation should be overridden.
	type	menu	A navigation node that could potentially contain a mix of any types of item. Most commonly will only contain link or audio items.
		contextMenu	A list of items of the specified type.
		artists	
		composers	
		albums	

Element	Attribute (and values)		Description
		playlists	
		tracks	
		genres	
		sections	Alphabetic sections.
		items	Generic result list. Most commonly a mix of menu nodes (type="link") and radio items (type="audio").
		folders	May contain subfolders, tracks and playlists entries.
<category>	text		Heading for the category.
	nextKey		A value to use for a key parameter to a /Browse request to get the next page of items for the category.
	parentKey		A value to use for a key parameter to a /Browse request to navigate back up the hierarchy if the default back navigation should be overridden.
<item>	type	link	A generic node in the browse hierarchy that leads to further nodes
		audio	A node that can be played directly
		artist	An item representing an artist
		composer	An item representing a composer
		album	An item representing an album or similar collection

Element	Attribute (and values)		Description
		playlist	An item representing a playlist or similar collection
		track	An item representing a single track
		text	A plain text node.
		section	An alphabetic section.
		folder	A folder in a folder browse.
	text		Main or first line of item description
	text2		Second line
	image		An icon or artwork for the item
	browseKey		A value to use for a key parameter to a subsequent /Browse request to descend the hierarchy.
	playURL		A URI that may be invoked directly to invoke the default play action for the item in question. Usually this is to add the item to the end of the play queue and start playing it.
	autoplayURL		A URI that may be invoked directly to add a track to the queue and play it and add subsequent tracks from the containing object (such as an album) to the auto-fill section of the play queue.
	contextMenuKey		A value to use for a key parameter to a /Browse request to obtain a result which is context-menu of actions related to the item.
	actionURL		A URI that may be invoked directly to perform the

Element	Attribute (and values)	Description
		specified action.

Context menu items may have the following values for the type attribute.

Attribute		Description
favourite	-add	Add the item as a favourite (or equivalent)
	-delete	Remove the item from the user's favourites
add		Add to the play queue
add	-now	Add to the play queue and play now
	-next	Add to the play queue after the current track
	-last	Add to the end of the play queue
addAll	-now	Add multi-track object to the play queue and play now
	-next	Add multi-track object to the play queue after the current track or multi-track object
	-last	Add multi-track object to the end of the play queue
playRadio		Play a radio station related to the item
delete		Delete the object (usually a playlist). User confirmation should be requested

Implementation Notes and Hints

The type attribute of an item is provided as a hint that may facilitate different display options.

The ability to browse the contents of an item is indicated by the presence of a browseKey attribute. The ability to play an (entire) item is indicated by the presence of a playURL (and possibly also autoplayURL) attribute. An item may have both a browseKey attribute and a playURL attribute.

When both playURL and autoplayURL attributes are available, which one to use as the default play option should be the subject of a user preference.

URI values will generally be relative URIs with an absolute path component. Relative URIs are resolved into absolute URIs according to RFC 3986.

browseKey, contextMenuKey and searchKey attribute values shall always be URI-encoded (percent escaped) when used as the value of a key parameter to a /Browse request, as shall any other request parameters.

When descending the browse hierarchy it may be useful for the UI page header to show some sort of breadcrumb(s), probably using the title (text) of the parent and grandparent nodes.

It may be useful to make available the context-menu for the parent when browsing its children.

It may be useful to consider the type of the parent when deciding how to display its children.

7.2 Search Music Content

Description

Command to search within a service.

Command

Browse?key=*key-value*&q=*searchText*

Parameters	Description
key	Value that is taken from a “searchKey” attribute value from an earlier response
q	The search string. Perform a search of the context specified by the key parameter (taken from a searchKey attribute from a response). In the absence of a key parameter, perform a top-level search.

Response

```
<browse sid="16" serviceIcon="/Sources/images/DeezerIcon.png" serviceName="Deezer"
service="Deezer" searchKey="Deezer:Search" type="menu">
    <item browseKey="/Artists?service=Deezer&expr=michael" text="Artists" type="link"/>
    <item browseKey="/Albums?service=Deezer&expr=michael" text="Albums" type="link"/>
    <item browseKey="/Songs?service=Deezer&expr=michael" text="Songs" type="link"/>
```

<item browseKey="/Playlists?service=Deezer&expr=michael" text="Playlists" type="link"/>

</browse>

Return the top level of search result. For further search result of Artists, Albums, Songs, or Playlists, a browse command with the “browseKey” as the key is required.

For example, to see the search result of Albums, send command:

http://192.168.1.100:11000/Browse?key=%2FAlbums%3Fservice%3DDeezer%26expr%3Dmichael

The response will be same as the response to normal /Browse command.

Response Attributes	Description
Refer to element/attribute tables in Browse command	

Example

http://192.168.1.100:11000/Browse?key=Deezer:Search&q=michael

Search for “michael” within the Deezer music service.

8. Player Grouping

This section describes commands for default player grouping and ungrouping. BluOS also supports fixed grouping, which is out of scope for this document.

BluOS uses the terminology primary player and secondary player. The primary player is the main player in the group. The primary player is used to select the music source. There is only one primary player. A secondary player is attached to the primary player. There can be multiple secondary players.

If a player is a secondary player then many requests, if directed to the secondary player, are internally proxied to the primary player. These include /Status, Playback Control, Play Queue Management and Content Browsing and Searching requests.

8.1 Group Two Players

Description

Group one secondary player to a primary player.

Request

/AddSlave?slave=*secondaryPlayerIP*&port=*secondaryPlayerPort*

Parameters	Description
slave	IP address of the secondary player.
port	Port number of the secondary player. The default port number is 11000. Players such as the NAD CI580, which has four players with one IP, use multiple ports.

Response

```
<addSlave>
    <slave port="11000" id="192.168.1.153"/>
</addSlave>
```

Response Attributes	Description
slave port	Port number of the secondary player that was just grouped.
Id	ID of the secondary player that was just grouped.

Example

http://192.168.1.100:11000/AddSlave?slave=192.168.1.153&port=11000

This groups player 192.168.1.153 to player 192.168.1.100. Player 192.168.1.100 is the primary player.

8.2 Add Multiple Players To A Group

Description

Group two or more players to a primary player.

Request

/AddSlave?slaves=*secondaryPlayerIPs*&ports=*secondaryPlayerPorts*

Parameters	Description
slaves	IP addresses of the secondary players to be added to the primary player. IP addresses are comma separated.
ports	The ports of the secondary players to be added to the primary player. Port numbers are comma separated.

Response

```
<addSlave>
    <slave port="11000" id="192.168.1.153"/>
    <slave port="11000" id="192.168.1.120"/>
</addSlave>
```

Response Attributes	Description
port	The port of the secondary player that was grouped.
Id	The id of the secondary player that was grouped.

Example

http://192.168.1.100:11000/AddSlave?slaves=192.168.1.153,192.168.1.120&ports=11000,11000

Groups secondary players 192.168.1.153 and 192.168.1.120 to primary player 192.168.1.100.

8.3 Remove One Player From A Group

Description

Remove a player from a group. If removing a secondary player from a group, the secondary player is ungrouped. If removing the primary player from a group of 3 or more players, the primary player is ungrouped and the remaining secondary players form a new group.

Request

/RemoveSlave?slave=*secondaryPlayerIP*&port=*secondaryPlayerPort*

Parameters	Description
slave	IP of the player (secondary) to be added to another player (primary).
port	Port of the player (secondary) to be added to another player (primary).

Response

<SyncStatus icon="/images/players/P300_nt.png" volume="4" modelName="PULSE" name="PULSE-0278" model="P300" brand="Bluesound" etag="25" outlevel="-62.9" schemaVersion="25" initialized="true" group="PULSE-0278+POWERNODE-0A6A" syncStat="25" id="192.168.1.100:11000"mac="90:56:82:9F:02:78">

<slave port="11000" id="192.168.1.120"/>

</SyncStatus>

Response Attributes	Description
	See /SyncStatus for details.

Example

http://192.168.1.100:11000/AddSlave?slave=192.168.1.153&port=11000

Ungroups player 192.168.1.153 from the group that has primary player 192.168.1.100

8.4 Remove Multiple Players From A Group

Description

Remove two or more players from a group.

Request

/RemoveSlave?slaves=*secondaryPlayerIPs*&ports=*secondaryPlayerPorts*

Parameters	Description
slaves	IP addresses of the secondary players to be removed from the primary player. IP addresses are comma separated.
ports	The ports of the secondary players to be removed from the primary player. Port numbers are comma separated.

Response

<SyncStatus icon="/images/players/P300_nt.png" volume="4" modelName="PULSE" name="PULSE-0278" model="P300" brand="Bluesound" etag="41" outlevel="-62.9" schemaVersion="25" initialized="true" syncStat="41" id="192.168.1.100:11000" mac="90:56:82:9F:02:78"></SyncStatus>

Response Attributes	Description
	See /SyncStatus for details.

Example

http://192.168.1.100:11000/RemoveSlave?slaves=192.168.1.153,192.168.1.120&ports=11000,11000

Removes players 192.168.1.153 and 192.168.1.120 from the group with primary player 192.168.1.100.

9. Player Reboot

This section describes command for player soft reboot.

9.1 Reboot A Player

Description

Soft reboot a player.

Request

POST command /reboot with parameter yes (any value)

Parameters	Description
yes	Any value (e.g. 1).

Response

Settings Updated

Rebooting. Please close this window.

Please wait...

Example

curl -d yes=1 192.168.1.100/reboot

10. Doorbell Chimes

This section describes command for player doorbell chime.

10.1 Doorbell Chimes

Description

Activate doorbell chimes.

Request

http://PLAYERIP:PORT/Doorbell?play=1

Parameters	Description
play	Play doorbell (always 1)

Response

<status enable="1" volume="38" chime="Doorbell:audio/chime_1.mp3"/>

Response Attributes	Description
enable	Indicate the chime
volume	Chime volume
chime	Chime audio

Example

http://192.168.1.100:11000/Doorbell?play=1

play doorbell chime

11. Direct Input

This section describes command for direct input source selection.

11.1 Direct Input Selection

Description

Direct input source selection.

Request

/Play?url=URL_value

Parameters	Description
url	The URL attribute from the response to /RadioBrowse?service=Capture

Response

<state>stream</state>

Response Attributes	Description
state	Indicate that the input is playing

Example

Step 1. Get the URL_value for parameter url

Request: http://192.168.1.100:11000/RadioBrowse?service=Capture

Response:

<radiotime service="Capture">
<item playerName="Tick
Tick" text="Bluetooth" inputType="bluetooth" id="input2" URL="Capture%3Abluez%3Abluetooth" image="
/images/BluetoothIcon.png" type="audio"/>
<item playerName="Tick Tick" text="Analog
Input" inputType="analog" id="input0" URL=**"Capture%3Aplughw%3Aimxnadadc%2C0%2F48000%2F
24%2F2%3Fid%3Dinput0"** image="/images/capture/ic_analoginput.png" type="audio"/>
<item playerName="Tick Tick" text="Optical
Input" inputType="spdif" id="input1" URL="Capture%3Ahw%3Aimxspdif%2C0%2F1%2F25%2F2%3Fid%
3Dinput1" image="/images/capture/ic_opticalinput.png" type="audio"/>
<item playerName="Tick
Tick" text="Spotify" id="Spotify" URL="Spotify%3Aplay" image="/Sources/images/SpotifyIcon.png" service
Type="CloudService" type="audio"/>
<category icon="/images/CB130Icon.png" text="**Test Hub**">
<remoteitem playerName="Test Hub" text="Analog Input" inputType="analog" id="hub-
192168114911000-

```
        input0" URL="Hub%3A%2F%2F192.168.1.149%3A11000%2Finput0" image="/images/capture/i
        c_analoginput.png" type="audio"/>
                <remoteitem playerName="Test Hub" text="Coaxial Input" inputType="spdif" id="hub-
        192168114911000-
        input3" URL="Hub%3A%2F%2F192.168.1.149%3A11000%2Finput3" image="/images/capture/ic
        _opticalinput.png" type="audio"/>
                <remoteitem playerName="Test Hub" text="HDMI ARC" inputType="arc" id="hub-
        192168114911000-
        input4" URL="Hub%3A%2F%2F192.168.1.149%3A11000%2Finput4" image="/images/capture/ic
        _tv.png" type="audio"/>
                <remoteitem playerName="Test Hub" text="Optical Input" inputType="spdif" id="hub-
        192168114911000-
        input2" URL="Hub%3A%2F%2F192.168.1.149%3A11000%2Finput2" image="/images/capture/ic
        _opticalinput.png" type="audio"/>
                <remoteitem playerName="Test Hub" text="Phono Input" inputType="phono" id="hub-
        192168114911000-
        input1" URL="Hub%3A%2F%2F192.168.1.149%3A11000%2Finput1" image="/images/capture/ic
        _vinyl.png" type="audio"/>
        </category>
</radiotime>
```

Step 2. Play Analog Input on the player

http://192.168.1.100:11000/Play?url=Capture%3Aplughw%3A2%2C0%2F48000%2F24%2F2%3Fid%3Dinput0

or play Analog Input of a HUB named “Test Hub”

http://192.168.1.100:11000/Play?url= Hub%3A%2F%2F192.168.1.149%3A11000%2Finput0

Note: Make sure the sources are connected and not hidden.

12. Bluetooth

This section describes command to change Bluetooth mode.

12.1 Change Bluetooth Mode

Description

Change Bluetooth mode: Manual, Automatic, Guest, Disabled.

Request

/audiomodes?bluetoothAutoplay=*value*

Parameters	Description
bluetoothAutoplay	Bluetooth mode *value* 0 means Manual, 1 means Automatic, 2 means Guest, 3 means Disabled.

No response

Example

Request: http://192.168.1.100:11000/audiomodes?bluetoothAutoplay=3

to disable bluetooth

13. Appendix

13.1 Lenbrook Service Discovery Protocol

Introduction

Popular discovery methods such as mDNS and SSDP use and rely on UDP multicast communication. Most current Lenbrook products use mDNS for discovery. Unfortunately we have found that a significant number of our customers have home networks where multicast traffic does not function correctly and our devices cannot be discovered reliably. This has resulted in many product returns and complaints from our distributors.

To address this issue we have created a custom discovery protocol called LSDP which utilizes UDP broadcast. Initial testing has shown this to be much more reliable than mDNS based discovery.

Protocol Overview

One goal of this protocol is to be relatively simple. It may be used in embedded devices with very limited memory.

The protocol uses all UDP broadcast packets to and from UDP port 11430. This port has been registered with the IANA and is assigned to Lenbrook for LSDP use as of March 27, 2014.

At steady state each node with a service to advertise broadcasts an Announce message approximately every minute.

At start up and when the service list or network parameters change seven packets shall be broadcast with short intervals to allow initial discovery and changes to propagate more quickly. For nodes advertising services these initial seven packets shall include an Announce message. For nodes trying to discover services the initial seven packets shall include a Query message. For services that are no longer available the seven packets should include a Delete message.

These initial packets are sent seven times due to the unreliable nature of UDP packets. In the unlikely case all seven packets fail services will still be discovered after a while from the one minute periodic Announce messages.

If a node receives a Query message for a service class it is advertising it shall respond with an Announce message after a short random time delay and reset its current Announce timeout.

The packet header and all message blocks include length fields. This gives extra flexibility and allows for backwards compatible changes to be made in the future. Extra fields or message types could be added in the future which older implementations could skip over when parsing. If we decided to make a backwards incompatible change there is also a version field in the packet header which can be incremented.

The protocol also allows for TXT records to be included with service advertisements similar to TXT records used with mDNS. This gives significant flexibility for additional arbitrary meta data to be included with service advertisements without changing the protocol.

Protocol Details

Timing

All packets sent should be scheduled with random timing or delays to help avoid collisions.

- Startup Packet Timing: 7 Packets at time = [0, 1, 2, 3, 5, 7, 10s] + (0 to 250ms random). These are absolute times, not delays. All 7 packets should be sent within about 10 seconds.
- Main Announce Period: 57s + (0 to 6s random)
- Query Response Delay: (0 to 750ms random)

Node ID

Each node shall have a unique ID which can be used to identify the node. The unique ID is included in Announce and Delete messages. Clients can use this value as a primary key when caching values and to uniquely identify a node. This unique ID can be a MAC address but should be the same for each interface if a node has multiple interfaces it is advertising on.

Packet Structure

Each packet starts with a packet header followed by an arbitrary number of message blocks. Each message block starts with a length field so unrecognized messages can be skipped. Unless otherwise specified all multi-byte number values shall be stored big endian (most significant bytes first). Unless otherwise specified all numbers are unsigned values. For example a one byte length can have the values 0 to 255.

Packet Header

Field	Bytes	Description
Length	1	Total length of header including this field.
Magic Word	4	This field shall be the four ASCII bytes of "LSDP". This helps us identify packets as being intended for use so we do not need to try to parse random data from some unexpected source.
Protocol Version	1	Version of protocol. If future changes are made to the protocol that is backwards incompatible this version

Field	Bytes	Description
		will be changed. The current version is 1.

Query Message

Field	Bytes	Description
Length	1	Total length of message including this field.
Message Type	1	"Q" = 0x51: Standard query for broadcast response. "R" = 0x52: Query for unicast response.
Count	1	Number of classes to query.
Class 1	2	16 bit (2 byte) class identifier.
...		Repeat previous field for each additional class.

Announce Message

Announce Header

Field	Bytes	Description
Length	1	Total length of message including complete announce header and announce records.
Message Type	1	"A" = 0x41
Node ID Length	1	Length of Node ID field.
Node ID	Variable	Unique Node ID of node sending the announcement. This is usually the MAC address of one of the nodes interfaces.
Address Length	1	Length of Address field. For IPv4 this shall be 4.

Field	Bytes	Description
Address	Variable	IP address of node.
Count	1	Number of announce records to follow.

Announce Record

Field	Bytes	Description
Class	2	16 bit (2 byte) class identifier.
TXT Count	1	Number of TXT records to follow. If zero the following fields are omitted.
Key 1 Length	1	Length of key name.
Key 1	Variable	Key name.
Value 1 Length	1	Length of value text.
Value 1	Variable	Value text.
...		Repeat previous 4 fields for each additional TXT record.

Delete Message

Field	Bytes	Description
Length	1	Total length of message including this field.
Message Type	1	"D" = 0x44
Node ID Length	1	Length of Node ID field.
Node ID	Variable	Unique Node ID of node sending the message. This is usually the MAC address of one of the nodes

Field	Bytes	Description
		interfaces.
Count	1	Number of classes to follow.
Class 1	2	16 bit (2 byte) class identifier.
...		Repeat previous field for each additional class.

Class ID Assignments

Class ID	Description	mDNS Equivalent
0x0001	BluOS Player	_musc._tcp
0x0002	BluOS Server	_muss._tcp
0x0003	BluOS Player (secondary in multi-zone players such as the CI580)	_musp._tcp
0x0004	sovi-mfg used for manufacturing testing.	_sovi-mfg._tcp
0x0005	sovi-keypad	_sovi-keypad._tcp
0x0006	BluOS Player (pair slave)	_musz._tcp
0x0007	Remote Web App (AVR OSD Web Page)	_remote-web-ui._tcp
0x0008	BluOS Hub	_mush._tcp
0xFFFF	All Classes - Can be used with Query Message.	

Note 1:

The overall LSDP packet needs to be treated as binary data.

Note 2:

If one Announcement Message can’t hold all nodes info (especially CI580), it will split into 2 or more Announcement Messages with each Message containing Header and Record and each Message hold whole node(s)’s info.