The protocol is provided as is.

Please note that it is solely intended for professional use. We always recommend to use the latest version of the protocol as commands may have changed, modified or added. The latest version can be downloaded here:


In general, Sound United will not provide any support for programming or using the commands in this document.

© Sound United LLC
HEOS CLI Protocol Specification

1. Overview
   1.1 Supported music services

2. Connection
   2.1 Controller Design Guidelines
      2.1.1 Driver Initialization
      2.1.2 Caveats
         2.1.2.1 Compatibility
         2.1.2.2 Issues & Solutions
      2.1.3 Miscellaneous

3. Command and Response Overview
   3.1 Commands
   3.2 Responses

4. Command and Response Details
   4.1 System Commands
      4.1.1 Register for Change Events
      4.1.2 HEOS Account Check
      4.1.3 HEOS Account Sign In
      4.1.4 HEOS Account Sign Out
      4.1.5 HEOS System Heart Beat
      4.1.6 HEOS Speaker Reboot
      4.1.7 Prettify JSON response
   4.2 Player Commands
      4.2.1 Get Players
      4.2.2 Get Player Info
      4.2.3 Get Play State
      4.2.4 Set Play State
      4.2.5 Get Now Playing Media
      4.2.6 Get Volume
      4.2.7 Set Volume
      4.2.8 Volume Up
      4.2.9 Volume Down
      4.2.10 Get Mute
      4.2.11 Set Mute
      4.2.12 Toggle Mute
      4.2.13 Get Play Mode
      4.2.14 Set Play Mode
      4.2.15 Get Queue
      4.2.16 Play Queue Item
      4.2.17 Remove Item(s) from Queue
      4.2.18 Save Queue as Playlist
      4.2.19 Clear Queue
      4.2.20 Move Queue
      4.2.21 Play Next
      4.2.22 Play Previous
      4.2.23 Set QuickSelect [LS AVR Only]
      4.2.24 Play QuickSelect [LS AVR Only]
      4.2.25 Get QuickSelects [LS AVR Only]
      4.2.26 Check for Firmware Update
   4.3 Group Commands
      4.3.1 Get Groups
      4.3.2 Get Group Info
      4.3.3 Set Group
      4.3.4 Get Group Volume
      4.3.5 Set Group Volume
      4.2.6 Group Volume Up
      4.2.7 Group Volume Down
      4.3.8 Get Group Mute
      4.3.9 Set Group Mute
      4.3.10 Toggle Group Mute
   4.4 Browse Commands
      4.4.1 Get Music Sources
      4.4.2 Get Source Info
      4.4.3 Browse Source
      4.4.4 Browse Source Containers
      4.4.5 Get Source Search Criteria
      4.4.6 Search
      4.4.7 Play Station
- 4.4.8 Play Preset Station
- 4.4.9 Play Input source
  - Limitations for the system when used multi devices.
- 4.4.10 Play URL
- 4.4.11 Add Container to Queue with Options
- 4.4.12 Add Track to Queue with Options
- 4.4.13 Get HEOS Playlists
- 4.4.14 Rename HEOS Playlist
- 4.4.15 Delete HEOS Playlist
- 4.4.16 Get HEOS History
- 4.4.17 Retrieve Album Metadata
- 4.4.18 Get Service Options for now playing screen - OBSOLETE
- 4.4.19 Set service option

5. Change Events (Unsolicited Responses)
- 5.1 Sources Changed
- 5.2 Players Changed
- 5.3 Group Changed
- 5.4 Player State Changed
- 5.5 Player Now Playing Changed
- 5.6 Player Now Playing Progress
- 5.7 Player Playback Error
- 5.8 Player Queue Changed
- 5.9 Player Volume Changed
- 5.10 Player Repeat Mode Changed
- 5.11 Player Shuffle Mode Changed
- 5.12 Group Volume Changed
- 5.13 User Changed

6. Error Codes
- 6.1 General Error Response
- 6.2 Error Code description

<table>
<thead>
<tr>
<th>Version</th>
<th>HEOS Version</th>
<th>Modifications</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.280.96</td>
<td>Initial release.</td>
<td>12/20/2014</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.1</td>
<td>1.304.61</td>
<td>Add set service option command.</td>
<td>05/27/2015</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.2</td>
<td>1.310.170</td>
<td>Remove support for play url. Special characters ('&amp;','=' , and '%') are encoded.</td>
<td>08/06/2015</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.3</td>
<td>1.331.70</td>
<td>Add reboot command. Support Tidal/SoundCloud /Amazon Music. Extend get_search_criteria to indicate if Play-All option is supported on searched tracks. Ability to create new station through Artist/Show /Track name. Add service specific transport control option list.</td>
<td>12/03/2015</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.4</td>
<td>1.331.120</td>
<td>Bug fixes. Documentation changes: Add transport control options for Amazon Music. Known Issues: Range queries doesn't work on Amazon Music.</td>
<td>01/21/2016</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>Version</td>
<td>Change Code</td>
<td>Description</td>
<td>Date</td>
<td>Author</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1.5</td>
<td>1.349.101</td>
<td>Add preset command to play stations from HEOS Favorites. Add players network connection type in get_players and get_player_info command responses. Fix issue with range queries on Amazon Music. Add issues and solutions section. Refer Issues &amp; Solutions. Remove support for Rdio as it is gone.</td>
<td>04/13/2016</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.6</td>
<td>1.364.110</td>
<td>Add limitations while using inputs. Add AVR inputs list.</td>
<td>07/25/2016</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.7</td>
<td>1.373.100</td>
<td>Add Source id for each Music service and source. Remove unused change events: source_data_changed, group_changed, player_mute_changed, group_mute_changed.</td>
<td>09/21/2016</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.8</td>
<td>1.397.190</td>
<td>Add support for Juke music service. Add support for adding station to HEOS Favorites from browse menu. Expand Thumbs Up/Down option to more music services. [LS AVR only] Add new commands set_quickselect, play_quickselect, and get_quickselects.</td>
<td>04/12/2017</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.9</td>
<td>1.406.140</td>
<td>Add support to Play Url</td>
<td>05/23/2017</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.10</td>
<td>1.430.160</td>
<td>Add check_update command. Add new response field 'serial' to get_players and get_player_info command responses. Add new response fields 'available' and 'service_username' to get_music_sources and get_source_info command responses. Add QQMusic to the supported music service list.</td>
<td>11/20/2017</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.11</td>
<td>1.442.150</td>
<td>Add new field value (unknown) to network connection types. Clarify on 'available' field in 'Get Music Sources' and 'Get Source Info' command response.</td>
<td>02/15/2018</td>
<td>Prakash Mortha</td>
</tr>
<tr>
<td>1.12</td>
<td>1.442.150</td>
<td>Document more inputs</td>
<td>05/14/2018</td>
<td>Prakash Mortha</td>
</tr>
</tbody>
</table>
1. Overview

The Denon HEOS is a network connected, wireless, multi-room music system. The HEOS Command Line Interface (CLI) allows external control systems to manage, browse, play, and get status from the Denon HEOS products. The HEOS CLI is accessed through a telnet connection between the HEOS product and the control system. The control system sends commands and receives responses over the network connection. The CLI commands and responses are in human readable (ascii) format. The command is a text string and the responses are in JSON format. The commands and responses for browsing music servers and services use a RESTFUL like approach while other commands and responses are more static.

1.1 Supported music services

Following table list out all supported online music services through HEOS. Please note, currently not all services are supported through CLI.

<table>
<thead>
<tr>
<th>Source ID (sid)</th>
<th>Service Name</th>
<th>Browse through CLI</th>
<th>Search/New station through CLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pandora</td>
<td>Yes</td>
<td>Yes (Create New Station)</td>
</tr>
<tr>
<td>2</td>
<td>Rhapsody</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>TuneIn</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Spotify</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Deezer</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Napster</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>iHeartRadio</td>
<td>Yes</td>
<td>Yes (Create New Station)</td>
</tr>
<tr>
<td>8</td>
<td>Sirius XM</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Soundcloud</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Tidal</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Future service</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>12</td>
<td>Rdio</td>
<td>Not supported in HEOS</td>
<td>Not supported in HEOS</td>
</tr>
<tr>
<td>13</td>
<td>Amazon Music</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>Future service</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>Moodmix</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>16</td>
<td>Juke</td>
<td>Not supported in HEOS</td>
<td>Not supported in HEOS</td>
</tr>
<tr>
<td>17</td>
<td>Future service</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>18</td>
<td>QQMusic</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Following table list out other supported music sources through CLI.

<table>
<thead>
<tr>
<th>Source ID (sid)</th>
<th>Source name</th>
<th>Browse supported</th>
<th>Search supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>Local USB Media/ Local DLNA servers</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>1025</td>
<td>HEOS Playlists</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
2. Connection

The HEOS products can be discovered using the UPnP SSDP protocol. Through discovery, the IP address of the HEOS products can be retrieved. Once the IP address is retrieved, a telnet connection to port 1255 can be opened to access the HEOS CLI and control the HEOS system. The HEOS product IP address can also be set statically and manually programmed into the control system. Search target name (ST) in M-SEARCH discovery request is 'urn:schemas-denon-com:device:ACT-Denon:1'.

The control system should use various Get commands to determine the players and groups currently in the HEOS system.

Controller software can control all HEOS speakers in the network by establishing socket connection with just one HEOS speaker. It is recommended not to establish socket connection to each HEOS speaker. This is to decrease network traffic caused by establishing socket connection to each HEOS speaker. Controller software can open multiple socket connections to the single HEOS speaker. Typically controllers will use one connection to listen for change events and one to handle user actions.

2.1 Controller Design Guidelines

2.1.1 Driver Initialization

In order to reduce number of UPnP devices running on the network, HEOS Speaker runs CLI module in a dormant mode. HEOS speaker spawns CLI core modules when the controller establishes the first socket connection to the speaker. What it all means for controller?

- Inability of CLI module to process player commands. This is because, by nature of UPnP, CLI module need some time to discover all players before they can be identified by their unique Id (pid)
- Spew of events when controller initially connects to the speaker. In order to avoid excessive event handling in a event driven controller system, the following initialization sequence is suggested:
  1. Un-register for change events. By default speaker doesn't send unsolicited events but still it is a good idea to send un-register command. This is done through 'register_for_change_events' command.
  2. If user credentials are available, sign-in to HEOS user account. This is done through 'sign_in' command.
  3. Retrieve current HEOS ecosystem status. This is done through commands like 'get_players', 'get_sources', 'get_groups', 'get_queue', 'get_now_playing_media', 'get_volume', 'get_play_state' etc.
  4. Register for change events. This is done through 'register_for_change_events' command.

If controller design involves disconnect and reconnect to HEOS speakers through CLI, it is recommended to keep a idle connection to HEOS Speaker thus avoiding CLI module to set back to dormant mode.

2.1.2 Caveats

2.1.2.1 Compatibility

Please take a look at the following suggestions to avoid breaking controller code due to future enhancements

- The 'message' field part of HEOS response is a string. The attribute value pair in this message string is delimited by ' & '. Further the attribute name and value is separated by '=' sign. Please note that new arguments can be added in the future.
- New JSON objects may be added to the 'payload' as part of future enhancements.

2.1.2.2 Issues & Solutions

Changes made to HEOS user account, through HEOS app will not reflect through CLI until the controller is restarted. Ex: Adding or removing music services to HEOS user account, through HEOS app will not reflect in get_music_sources command response until the controller is restarted.

Solution: Controller needs to re sign-in to HEOS account to reflect changes made through HEOS app, with out restarting the controller. So, in addition to performing HEOS account sign-in as part of driver initialization process, it is highly recommended to provide sign-out and sign-in option through end users UI screen. End user need to re-signIn when he adds/removes music service through HEOS app.
2.1.3 Miscellaneous

- Controllers can add custom argument `SEQUENCE=<number>` in browse commands to associate command and response. This is possible because the 'message' field in the response packet includes all the arguments sent in the command. Please let us know if you need additional custom argument other than 'SEQUENCE'. This is to avoid accidentally using HEOS command arguments for special purpose.
- Maximum number of simultaneous socket connections supported by HEOS speaker is 32.
- Service specific transport control options are as follows:

<table>
<thead>
<tr>
<th>Services</th>
<th>Type</th>
<th>Supported Transport Controls by CU</th>
<th>Supported Transport Controls in HEOS App (No significance. Only for Reference)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Music</td>
<td>station</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>Deezer</td>
<td>station</td>
<td>Play, Pause, Stop, PlayNext</td>
<td>Play, Pause, PlayNext</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>iHeart Radio</td>
<td>station</td>
<td>Play, Stop</td>
<td>Play, Stop, Scan</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>Napster</td>
<td>station</td>
<td>Play, Pause, Stop, PlayNext</td>
<td>Play, Pause, PlayNext</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>Pandora</td>
<td>station</td>
<td>Play, Pause, Stop, PlayNext</td>
<td>Play, Pause, PlayNext</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Rhapsody</td>
<td>station</td>
<td>Play, Pause, Stop, PlayNext</td>
<td>Play, Pause, PlayNext</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>SoundCloud</td>
<td>station</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>SiriusXM</td>
<td>station</td>
<td>Play, Stop</td>
<td>Play, Stop</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Tidal</td>
<td>station</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>Tunein</td>
<td>station</td>
<td>Play, Stop</td>
<td>Play, Stop</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>Local Music</td>
<td>station</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>Favorites</td>
<td>station</td>
<td>*Depending on playing service</td>
<td>*Depending on playing service</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Playlists</td>
<td>station</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>History</td>
<td>station</td>
<td>*Depending on playing service</td>
<td>*Depending on playing service</td>
</tr>
<tr>
<td></td>
<td>song</td>
<td>Play, Pause, Stop, PlayNext, PlayPrevious</td>
<td>Play, Pause, PlayNext, PlayPrevious</td>
</tr>
<tr>
<td>AUX Input</td>
<td>station</td>
<td>Play, Stop</td>
<td>Play, Stop</td>
</tr>
</tbody>
</table>
3. Command and Response Overview

3.1 Commands

HEOS CLI commands are in the following general format:
heos://command_group/command?attribute1=value1&attribute2=value2&…&attributeN=valueN

Command string delimiter is "\r\n".

Note: Special characters, i.e '&', '=' and '%' in attribute/value needs to be encoded to '%26(&)', '%3D(=)', and '%25(%)'. Most of the time, controllers use the same string that is received in previous command response. For example, while preparing 'play_stream' /'add_to_queue' command, controllers will use the strings obtained in 'browse' command response. Those strings are already encoded. So, controllers are not required to perform any special action. However, controllers might need to decode the encoded strings before they can be properly displayed on the controller GUI.

3.2 Responses

The responses to commands are in JSON format and use the following general structure:
{
    "heos": {
        "command": "command_group'/command", "result": "success" or fail", "message": "other result information"
    },
    "payload": {
        'Rest of response data'
    }
}

Some command responses will not include a payload.
If the *result* of the command is *fail* then the *message* information contains the error codes for the failure. The error codes can be found in section 'Error Code description'.

Some commands will also cause unsolicited events. For example, sending the 'player/clear_queue' command will cause the Player Queue Changed event and could also cause the Player State Changed event.

When the actual response can't be populated immediately, a special response will be sent back as shown below. This usually occurs during browse/search as CLI needs to retrieve data from remote media server or online service.

{
    "heos": {
        "command": "command_group'/command", "result": "success", "message": "command under process"
    }
}

JSON command response delimiter is "\r\n".

Note: Special characters '&', '=' and '%' in the JSON response fields are encoded to '%26(&)', '%3D(=)', and '%25(%)'.

4. Command and Response Details

4.1 System Commands
4.1.1 Register for Change Events

By default HEOS speaker does not send Change events. Controller needs to send this command with enable=on when it is ready to receive unsolicit responses from CLI. Please refer to "Driver Initialization" section regarding when to register for change events.

Command: heos://system/register_for_change_events?enable='on_or_off'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Register or unregister for change events.</td>
<td>on,off</td>
</tr>
</tbody>
</table>

Response:

```
{
    "heos": {
        "command": "system/register_for_change_events",
        "result": "success",
        "message": "enable='on_or_off'"
    }
}
```

Example: heos://system/register_for_change_events?enable=on

4.1.2 HEOS Account Check

Command: heos://system/check_account

This command returns current user name in its message field if the user is currently singed in.

Response:

```
{
    "heos": {
        "command": "system/check_account",
        "result": "success",
        "message": "signed_out" or "signed_in&un=<current user name>"
    }
}
```

Example: heos://system/check_account

4.1.3 HEOS Account Sign In

Command: heos://system/sign_in?un=heos_username&pw=heos_password

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>un</td>
<td>HEOS account username</td>
<td>N/A</td>
</tr>
<tr>
<td>pw</td>
<td>HEOS account password</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

```
{
    "heos": {
        "command": "system/sign_in",
        "result": "success",
        "message": "signed_in&un=<current user name>"
    }
}
```

Example: heos://system/sign_in?un=user@gmail.com&pw=12345

4.1.4 HEOS Account Sign Out

Command: heos://system/sign_out

Response:

```
"heos": {
    "command": "system/sign_out",
    "result": "success",
    "message": "signed_out"
  }
}  
Example: heos://system/sign_out

4.1.5 HEOS System Heart Beat

Command: heos://system/heart_beat
Response:
  {
    "heos": {
      "command": "system/heart_beat",
      "result": "success",
      "message": ""
    }
  }
Example: heos://system/heart_beat

4.1.6 HEOS Speaker Reboot

Using this command controllers can reboot HEOS device. This command can only be used to reboot the HEOS device to which the controller is connected through CLI port.

Command: heos://system/reboot
Response:
  {
    "heos": {
      "command": "system/reboot",
      "result": "success",
      "message": ""
    }
  }
Example: heos://system/reboot

4.1.7 Prettify JSON response

Helper command to prettify JSON response when user is running CLI controller through telnet.

Command: heos://system/prettify_json_response?enable='on_or_off'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Enable or disable prettification of JSON response.</td>
<td>on, off</td>
</tr>
</tbody>
</table>

Response:
  {
    "heos": {
      "command": "system/prettify_json_response",
      "result": "success",
      "message": "enable='on_or_off'"
    }
  }
Example: heos://system/prettify_json_response?enable=on

4.2 Player Commands

4.2.1 Get Players
Command: heos://player/get_players

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id</td>
<td>N/A</td>
</tr>
<tr>
<td>gid</td>
<td>pid of the Group leader</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| network   | Network connection type | | • wired  
                   • wifi  
                   • unknown (not applicable for external controllers) |
| lineout   | LineOut level type | 1 - variable  
                   2 - Fixed |
| control   | Only valid when lineout level type is Fixed (2). | 1 - None  
                   2 - IR  
                   3 - Trigger  
                   4 - Network |
| serial    | Only listed if device has valid serial number | N/A         |

Note: The group id field (gid) is optional. The 'gid' field will only be appeared if the player(s) is part of a group.
Note: control field is only populated when lineout level type is Fixed (lineout = 2)

Response:
```
{
  "heos": {
    "command": "player/get_players",
    "result": "success",
    "message": ""
  },
  "payload": [
    {
      "name": "player name 1",
      "pid": "player id 1",
      "gid": "group id",
      "model": "player model 1",
      "version": "player version 1",
      "network": "wired",
      "lineout": "level type",
      "control": "control option",
      "serial": "serial number"
    },
    {
      "name": "player name 2",
      "pid": "player id 2",
      "gid": "group id",
      "model": "player model 2",
      "version": "player version 2",
      "network": "wifi",
      "lineout": "level type",
      "control": "control option",
      "serial": "serial number"
    },
    ...
  ]
}
```
4.2.2 Get Player Info

Command: heos://player/get_player_info?pid=player_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>gid</td>
<td>pid of the Group leader</td>
<td>N/A</td>
</tr>
<tr>
<td>network</td>
<td>Network connection type</td>
<td>wired, wifi, unknown (not applicable for external controllers)</td>
</tr>
<tr>
<td>lineout</td>
<td>LineOut level type</td>
<td>1 - variable, 2 - Fixed</td>
</tr>
<tr>
<td>control</td>
<td>Only valid when lintout level type is Fixed (2).</td>
<td>1 - None, 2 - IR, 3 - Trigger, 4 - Network</td>
</tr>
<tr>
<td>serial</td>
<td>Only listed if device has valid serial number</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note: The group id field (gid) is optional. The 'gid' field will only be appeared if the player(s) is part of a group. Note: control field is only populated when lineout level type is Fixed (lineout = 2)

Response:

```json
{
    "heos": {
        "command": "player/get_player_info",
        "result": "success",
        "message": "pid='player_id'"
    },
    "payload": {
        "name": "player name",
        "pid": "player id",
        "gid": "group id",
        "model": "player model",
        "version": "player version",
        "network": "wired",
        "lineout": "level type",
        "control": "control option",
        "serial": "serial number"
    }
}
```

Example: heos://player/get_player_info?pid=1

4.2.3 Get Play State

Command: heos://player/get_play_state?pid=player_id
### 4.2.4 Set Play State

Command: `heos://player/set_play_state?pid=player_id&state=play_state`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>state</td>
<td>Player play state</td>
<td>play, pause, stop</td>
</tr>
</tbody>
</table>

Response:
```
{  "heos": {    "command": "player/set_play_state ",    "result": "success",    "message": "pid='player_id'&state='play_state"  }
}
```

Example: `heos://player/set_play_state?pid=1&state=play`

Note: Play state of a group can be controlled by sending `set_play_state` command to any of the player in the group.

### 4.2.5 Get Now Playing Media

Command: `heos://player/get_now_playing_media?pid=player_id`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>id (options)</td>
<td>Options available for now playing media</td>
<td>Following options are currently supported for now playing media</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 - Thumbs Up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 - Thumbs Down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 - Add station to HEOS Favorites</td>
</tr>
</tbody>
</table>

Response:
The following response provides example when the speaker is playing a song.

**Note:** For local music and DLNA servers sid will point to Local Music Source id.
The following response provides example when the speaker is playing a station.

```
[ "heos": {
  "command": "player/get_now_playing_media",
  "result": "success",
  "message": "pid='player_id'"
},
  "payload": {
    "type": "song",
    "song": "song name",
    "album": "album name",
    "artist": "artist name",
    "image_url": "image url",
    "mid": "media id",
    "qid": "queue id",
    "sid": source_id
    "album_id": "Album Id"
  }
}
```

Example: heos://player/get_now_playing_media?pid=1

### 4.2.6 Get Volume

Command: heos://player/get_volume?pid='player_id'

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

```
[ "heos": {
  "command": "player/get_volume",
  "result": "success",
  "message": "pid='player_id'"
},
  "payload": {
    "type": "station",
    "song": "song name",
    "album": "album name",
    "artist": "artist name",
    "image_url": "image url",
    "mid": "media id",
    "qid": "queue id",
    "sid": source_id
  }
  "options": [
    { "play": [ "id": 19, "name": "Add to HEOS Favorites" ] }
  ]
}
```
4.2.7 Set Volume

Command: heos://player/set_volume?pid=player_id&level=vol_level

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>level</td>
<td>Player volume level</td>
<td>0 to 100</td>
</tr>
</tbody>
</table>

Response:
```
{"heos": {
  "command": "player/ set_volume",
  "result": "success",
  "message": "pid='player_id'&level='vol_level'"
}}
```

Example: heos://player/set_volume?pid=2&level=30

4.2.8 Volume Up

Command: heos://player/volume_up?pid=player_id&step=step_level

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>step</td>
<td>Player volume step level</td>
<td>1 to 10(default 5)</td>
</tr>
</tbody>
</table>

Response:
```
{"heos": {
  "command": "player/ volume_up",
  "result": "success",
  "message": "pid='player_id'&step='step_level'"
}}
```

Example: heos://player/volume_up?pid=2&step=5

4.2.9 Volume Down

Command: heos://player/volume_down?pid=player_id&step=step_level

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>level</td>
<td>Player volume step level</td>
<td>1 to 10(default 5)</td>
</tr>
</tbody>
</table>

Response:
```
{"heos": {
  "command": "player/ volume_down",
  "result": "success",
  "message": "pid='player_id'&step='step_level'"
}}
```

Example: heos://player/volume_down?pid=2&step=5
Example: heos://player/volume_down?pid=2&step=5

### 4.2.10 Get Mute

Command: heos://player/get_mute?pid=player_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:
```
{  
  "heos": {  
    "command": " player/ get_mute ",  
    "result": "success",  
    "message": "pid='player_id'&state='on_or_off'"  
  }  
}
```

Example: heos://player/get_mute?pid=1

### 4.2.11 Set Mute

Command: heos://player/set_mute?pid=player_id&state=on_or_off

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>state</td>
<td>Player mute state</td>
<td>on, off</td>
</tr>
</tbody>
</table>

Response:
```
{  
  "heos": {  
    "command": " player/ set_mute ",  
    "result": "success",  
    "message": "pid='player_id'&state='on_or_off'"  
  }  
}
```

Example: heos://player/set_mute?pid=3&state=off

### 4.2.12 Toggle Mute

Command: heos://player/toggle_mute?pid=player_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:
```
{  
  "heos": {  
    "command": " player/ toggle_mute ",  
    "result": "success",  
    "message": "pid='player_id'&state='on_or_off'"  
  }  
}
```
4.2.13 Get Play Mode

Command: heos://player/get_play_mode?pid=player_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

```json
{
  "heos": {
    "command": "player/get_play_mode",
    "result": "success",
    "message": "pid='player_id'&&repeat=on_all_or_on_one_or_off&shuffle=on_or_off"
  }
}
```

Example: heos://player/get_play_mode?pid=1

4.2.14 Set Play Mode

Command: heos://player/set_play_mode?pid='player_id'&repeat=on_all_or_on_one_or_off&shuffle=on_or_off

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>repeat</td>
<td>Player repeat state</td>
<td>on_all, on_one, off</td>
</tr>
<tr>
<td>shuffle</td>
<td>Player shuffle state</td>
<td>on, off</td>
</tr>
</tbody>
</table>

Response:

```json
{
  "heos": {
    "command": "player/set_play_mode",
    "result": "success",
    "message": "pid='player_id'&repeat=on_all_or_on_one_or_off&shuffle=on_or_off"
  }
}
```

Example: heos://player/set_play_mode?pid=1&repeat=on_all&shuffle=off

4.2.15 Get Queue

Command: heos://player/get_queue?pid=player_id&range=start#, end#

Range is start and end record index to return. Range parameter is optional. Omitting range parameter returns all records but a maximum of 100 records are returned per response.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Range is start and end record index to return. Range parameter is optional. Omitting range parameter returns all records up to a maximum of 100 records per response.

**Response:**

```json
{
  "heos": {
    "command": "player/get_queue",
    "result": "success",
    "message": "pid=player_id&range=start#, end#"
  },
  "payload": [
    {
      "song": "song name 1",
      "album": "album name 1",
      "artist": "artist name 1",
      "image_url": "image_url 1",
      "qid": "queue id 1",
      "mid": "media id 1"
    },
    {
      "song": "song name 2",
      "album": "album name 2",
      "artist": "artist name 2",
      "image_url": "image_url 2",
      "qid": "queue id 2",
      "mid": "media id 2"
    },
    {
      "song": "song name N",
      "album": "album name N",
      "artist": "artist name N",
      "image_url": "image_url N",
      "qid": "queue id N",
      "mid": "media id N"
    }
  ]
}
```

Example: `heos://player/get_queue?pid=1&range=0,10`

### 4.2.16 Play Queue Item

**Command:** `heos://player/play_queue?pid=player_id&qid=queue_song_id`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>qid</td>
<td>Queue id for song returned by 'get_queue' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Response:**

```json
[
  "heos": {
```
4.2.17 Remove Item(s) from Queue

Command: heos://player/remove_from_queue?pid=player_id&qid=queue_id_1,queue_id_2,…,queue_id_n

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>qid</td>
<td>List of comma separated queue_id's where each queue id for song is returned by 'get_queue' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:
{
    "heos": {
        "command": "player/remove_from_queue",
        "result": "success",
        "message": "pid='player_id'&qid=queue_id_1,queue_id_2,…,queue_id_n"
    }
}

Example: heos://player/remove_from_queue?pid=1&qid=4,5,6

4.2.18 Save Queue as Playlist

Command: heos://player/save_queue?pid=player_id&name=playlist_name

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>name</td>
<td>String for new playlist name limited to 128 unicode characters</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:
{
    "heos": {
        "command": "player/save_queue",
        "result": "success",
        "message": "pid='player_id'&name='playlist_name"
    }
}

Example: heos://player/save_queue?pid=1&name=great playlist

4.2.19 Clear Queue

Command: heos://player/clear_queue?pid=player_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>
4.2.20 Move Queue

Command: heos://player/move_queue_item?pid=player_id&sqid=source_queue_id_1,source_queue_id_2,...,source_queue_id_n&dqid=destination_queue_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>sqid</td>
<td>List of comma separated queue_id's where each queue id for song is returned by 'get_queue' command</td>
<td>list item range: 1 to size of queue</td>
</tr>
<tr>
<td>dqid</td>
<td>User select this value as the destination of contents which is indicated in sqid.</td>
<td>1 to size of queue</td>
</tr>
</tbody>
</table>

Response:
{
  "heos": {
    "command": "player/move_queue_item ",
    "result": "success",
    "message": "pid='player_id'&sqid='source_queue_id_1',source_queue_id_2',...,source_queue_id_n',dqid='destination_queue_id"
  }
}

Example: heos://player/move_queue_item?pid=2,4&sqid=2&dqid=6

4.2.21 Play Next

Command: heos://player/play_next?pid=player_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:
{
  "heos": {
    "command": "player/play_next",
    "result": "success",
    "message": "pid=player_id"
  }
}

Example: heos://player/play_next?pid=1

4.2.22 Play Previous

Command: heos://player/play_previous?pid=player_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
</table>

Example: heos://player/play_previous?pid=1
4.2.23 Set QuickSelect [LS AVR Only]

Command: `heos://player/set_quickselect?pid=player_id&id=<quick select id>`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>id</td>
<td>Quick select Id to which currently playing source needs to be stored</td>
<td>1-6</td>
</tr>
</tbody>
</table>

Response:
```
{
  "heos": {
    "command": "player/set_quickselect",
    "result": "success",
    "message": "pid=player_id&id=<quick select id>"
  }
}
```

Example: `heos://player/set_quickselect?pid=1&id=2`

Currently supported HEOS products: LEGO AVR, HEOS BAR

4.2.24 Play QuickSelect [LS AVR Only]

Command: `heos://player/play_quickselect?pid=player_id&id=<quick select id>`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>id</td>
<td>Quick select Id whose source needs to be played</td>
<td>1-6</td>
</tr>
</tbody>
</table>

Response:
```
{
  "heos": {
    "command": "player/play_quickselect",
    "result": "success",
    "message": "pid=player_id&id=<quick select id>"
  }
}
```

Example: `heos://player/play_quickselect?pid=1&id=2`

Currently supported HEOS products: LEGO AVR, HEOS BAR

4.2.25 Get QuickSelects [LS AVR Only]

Command: `heos://player/get_quickselects?pid=player_id`

Command: `heos://player/get_quickselects?pid=player_id&id=<quick select id>`
### 4.2.26 Check for Firmware Update

**Command:** `heos://player/check_update?pid=player_id`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>update</td>
<td>Indicates if firmware update is available or not</td>
<td><code>update_none, update_exist</code></td>
</tr>
</tbody>
</table>

**Response:**

```
{
  "heos": {
    "command": "player/check_update",
    "result": "success",
    "message": "pid=player_id"
  },
  "payload": {
    "update": "update_none/update_exist",
  }
}
```

Example: `heos://player/check_update?pid=1`

Currently supported HEOS products: LEGO AVR, HEOS BAR

### 4.3 Group Commands
4.3.1 Get Groups

Command: heos://group/get_groups
Response:
{
  "heos": {
    "command": "player/get_groups",
    "result": "success",
    "message": ""
  },
  "payload": [
    {
      "name": "group name 1",
      "gid": "group id 1",
      "players": [
        {
          "name": "player name 1",
          "pid": "player id 1",
          "role": "player role 1 (leader or member)"
        },
        {
          "name": "player name 2",
          "pid": "player id 2",
          "role": "player role 2 (leader or member)"
        },
        ...
        {
          "name": "player name N",
          "pid": "player id N",
          "role": "player role N (leader or member)"
        }
      ]
    },
    {
      "name": "group name 2",
      "gid": "group id 2",
      "players": [
        {
          "name": "player name 1",
          "pid": "player id 1",
          "role": "player role 1 (leader or member)"
        },
        {
          "name": "player name 2",
          "pid": "player id 2",
          "role": "player role 2 (leader or member)"
        },
        ...
        {
          "name": "player name N",
          "pid": "player id N",
          "role": "player role N (leader or member)"
        }
      ]
    },
    ...
    {
      "name": "group name N",
      "gid": "group id N",
      "players": [
        {
          "name": "player name 1",
          "pid": "player id 1",
          "role": "player role 1 (leader or member)"
        }
      ]
    }
  ]
}
4.3.2 Get Group Info

Command: heos://group/get_group_info?gid=group_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:
```
[ {
    "heos": {
      "command": "player/get_groups",
      "result": "success",
      "message": "gid=group_id"
    },
    "payload": {
      "name": "group name 1",
      "gid": "group id 1",
      "players": [
        {
          "name": "player name 1",
          "pid": "player id 1",
          "role": "player role 1 (leader or member)"
        },
        {
          "name": "player name 2",
          "pid": "player id 2",
          "role": "player role 2 (leader or member)"
        },
        ...
      ]
    }
  },
  ...
  ]
```

Example: heos://group/get_group_info?gid=1

4.3.3 Set Group

This command is used to perform the following actions:
Create new group:

Creates new group. First player id in the list is group leader.
Ex: heos://group/set_group?pid=3,1,4

Modify existing group members:

Adds or delete players from the group. First player id should be the group leader id.
Ex: heos://group/set_group?pid=3,1,5

Ungroup all players in the group

Ungroup players. Player id (pid) should be the group leader id.
Ex: heos://group/set_group?pid=3

Command: heos://group/set_group?pid=player_id_leader, player_id_member_1,…,player_id_member_n

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>List of comma separated player_id's where each player id is returned by 'get_players' or 'get_groups' command; first player_id in list is group leader</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

The following response provides example when a group is created/modified.

```json

"heos": {
  "command": "player/set_group",
  "result": "success",
  "message": "gid='new group_id'&name='group_name'&pid='player_id_1, player_id_2,…,player_id_n'"
}
```

The following response provides example when all the speakers in the group are un-grouped.

```json

"heos": {
  "command": "player/set_group",
  "result": "success",
  "message": "pid='player_id'"
}
```

Example: heos://group/set_group?pid=3,1,4

### 4.3.4 Get Group Volume

Command: heos://group/get_volume?gid=group_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

```json

"heos": {
  "command": "group/get_volume",
  "result": "success",
  "message": "gid=group_id&level=vol_level"
}
```

Example: heos://group/set_group?pid=3,1,4
4.3.5 Set Group Volume

Command: heos://group/set_volume?gid=group_id&level=vol_level

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>level</td>
<td>Group volume level</td>
<td>0 to 100</td>
</tr>
</tbody>
</table>

Response:
```
{
  "heos": {
    "command": "group/set_volume",
    "result": "success",
    "message": "gid='group_id'&level='vol_level'"
  }
}
```

Example: heos://group/set_volume?gid=1&level=30

4.2.6 Group Volume Up

Command: heos://group/volume_up?gid=group_id&step=step_level

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>step</td>
<td>Group volume step level</td>
<td>1 to 10 (default 5)</td>
</tr>
</tbody>
</table>

Response:
```
{
  "heos": {
    "command": "group/volume_up",
    "result": "success",
    "message": "gid='group_id'&step='step_level'"
  }
}
```

Example: heos://group/volume_up?gid=1&step=5

4.2.7 Group Volume Down

Command: heos://group/volume_down?gid=group_id&step=step_level

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>level</td>
<td>Group volume step level</td>
<td>1 to 10 (default 5)</td>
</tr>
</tbody>
</table>

Response:
```
{
  "heos": {
    "command": "group/volume_down",
    "result": "success",
    "message": "gid='group_id'&step='step_level'"
  }
}
```
Example: heos://group/volume_down?gid=1&step=5

### 4.3.8 Get Group Mute

**Command:** heos://group/get_mute?gid=group_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Response:**
```json
{
   "heos": {
      "command": "group/get_mute",
      "result": "success",
      "message": "gid='group_id'&state='on_or_off'"
   }
}
```

Example: heos://group/get_mute?gid=1

### 4.3.9 Set Group Mute

**Command:** heos://group/set_mute?gid=group_id&state=on_or_off

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>state</td>
<td>Group mute state</td>
<td>on, off</td>
</tr>
</tbody>
</table>

**Response:**
```json
{
   "heos": {
      "command": "group/set_mute",
      "result": "success",
      "message": "gid=group_id&state=on_or_off"
   }
}
```

Example: heos://group/set_mute?gid=1&state=off

### 4.3.10 Toggle Group Mute

**Command:** heos://group/toggle_mute?gid=group_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>gid</td>
<td>Group id returned by 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Response:**
```json
{
   "heos": {
      "command": "group/toggle_mute",
      "result": "success",
      "message": "gid=group_id"
   }
}
```

Example: heos://group/toggle_mute?gid=1
Example: heos://group/toggle_mute?gid=1

### 4.4 Browse Commands

#### 4.4.1 Get Music Sources

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>available</td>
<td>Only valid for online music services. If true 'service_username' will provide user name of the service account. This should not be treated as the music service being supported through CLI or not. The services supported through CLI is listed in 'Supported music services' section.</td>
<td>N/A</td>
</tr>
<tr>
<td>service_username</td>
<td>Provides user name of the service account. Only available for online music services when 'available' field is set to true.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Command: heos://browse/get_music_sources
Response:
```
{
    "heos": {
        "command": "browse/get_music_sources",
        "result": "success",
        "message": ""
    },
    "payload": [
        {
            "name": "source name 1",
            "image_url": "source logo url 1",
            "type": "source type 1",
            "sid": source_id_1,
            "available": "true/false",
            "service_username": "user name of the service account"
        },
        {
            "name": "source name 2",
            "image_url": "source logo url 2",
            "type": "source type 2",
            "sid": source_id_2,
            "available": "true/false",
            "service_username": "user name of the service account"
        },
        {
            "name": "source name N",
            "image_url": "source logo url N",
            "type": "source type N",
            "sid": source_id_N,
            "available": "true/false",
            "service_username": "user name of the service account"
        }
    ]
}
```

Example: heos://browse/get_music_sources
The following are valid source types:

- music_service
- heos_service
- heos_server
- dlna_server

### 4.4.2 Get Source Info

Command: heos://browse/get_source_info?sid=source_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command (Or) Source id returned by 'browse' command when browsing source types 'heos_server' and 'heos_service'</td>
<td>N/A</td>
</tr>
<tr>
<td>available</td>
<td>Only valid for online music services. If true 'service_username' will provide user name of the service account. This should not be treated as the music service being supported through CLI or not. The services supported through CLI is listed in 'Supported music services' section.</td>
<td>N/A</td>
</tr>
<tr>
<td>service_username</td>
<td>Provides user name of the service account. Only available for online music services when 'available' field is set to true.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

```
{
  "heos": {
    "command": "browse/get_source_info",
    "result": "success",
    "message": ""
  },
  "payload": [
    {
      "name": "source name",
      "image_url": "source logo url",
      "type": "source type",
      "sid": source_id_1,
      "available": "true/false",
      "service_username": "user name of the service account"
    }
  ]
}
```

Example: heos://browse/get_source_info

The following are valid source types:

- music_service
- heos_service
- heos_server
- dlna_server
4.4.3 Browse Source

Command: heos://browse/browse?sid=source_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command (Or) Source id returned by 'browse' command when browsing source types 'heos_server' and 'heos_service'</td>
<td>N/A</td>
</tr>
<tr>
<td>id (options)</td>
<td>Options available for current browse level Following options are currently supported for 'Browse Source' command 13 - Create New Station (Pandora, iHeartRadio) 20 - Remove from HEOS Favorites (Favorites)</td>
<td></td>
</tr>
<tr>
<td>scid</td>
<td>criteria for creating new station Possibilities: 1 - Artist (default) (Criteria String: Create New Station by Artists) 5 - Show (Criteria String: Create New Station by Shows) 3 - Track (Criteria String: Create New Station by Tracks)</td>
<td></td>
</tr>
<tr>
<td>range</td>
<td>Range is start and end record index to return. Range parameter is optional. Omitting range parameter returns all records up to a maximum of either 50 or 100 records per response. The default maximum number of records depend on the service type. range starts from 0</td>
<td></td>
</tr>
</tbody>
</table>

This command is used under two scenarios.

Browsing actual media sources of type 'heos_server' and 'heos_service'.

The command 'Get Music Sources' lists all music servers (type 'heos_server') in the network under one virtual source called 'Local Music'. Other virtual source that represents all auxiliary inputs (type 'heos_service') is 'AUX Input'.

Browsing top music view. Results of this command depends on the music source selected.

Note: Optionally this command returns service 'options' that are available for current browse items. Please refer to 'Get Service Options for now playing screen' for service options available on now playing screen.

Note: The following response provides examples of the various service options. The actual response will depend on the service options available for a given source type.

Response while browsing actual media sources of type 'heos_server' and 'heos_service'. These includes 'Local Music', 'History', 'AUX Inputs', 'Playlists', and 'Favorites'.

```json
{
  "heos": {
    "command": "browse/browse",
    "result": "success",
    "message": "sid=source_id&returned=items_in_current_response&count=total_items_available"
  },
  "payload": {
    "name": "source name 1",
    "image_url": "source logo url 1",
    "sid": "source id 1"
  }
}
```
Example: heos://browse/browse?sid=1

Response when browsing top music view in an actual music server/music services.

**Note:** the following response provides examples of the various media types. The actual response will depend on the source browsed and the hierarchy supported by that source.

```
{  "heos": {    "command": "browse/browse",    "result": "success",    "message": "sid=source_id&returned=items_in_current_response&count=total_items_available"  },  "payload": [    {      "container": "yes",      "playable": "no",      "type": "artist",      "name": "artist name",      "image_url": "artist image url",      "cid": "container id",      "mid": "media id"    },    {      "container": "yes",      "playable": "yes",      "type": "album",      "name": "album name",      "image_url": "album image url",      "artist": "artist name",      "cid": "container id",      "mid": "media id"    },    {      "container": "no",      "playable": "yes",      "type": "song",      "name": "song name",      "image_url": "album image url",      "artist": "artist name",      "album": "album name",      "mid": "media id"    },    {      "container": "yes",      "playable": "yes",      "type": "playlist",      "name": "playlist name",      "image_url": "playlist image url",      "artist": "artist name",      "album": "album name",      "mid": "media id"    }  ]
```
*playable": "no",
*type": "container",
*name": "container name",
*image_url": "container image url",
*cid": "container id",
*mid": "media id"
},
{
*container": "no",
*playable": "yes",
*type": "station",
*name": "station name",
*image_url": "station url",
*mid": "media id"
}
],
"options": [
{
*browse": [
{
"id": 20,
"name": "Remove from HEOS Favorites"
}
]
}
]

Example: heos://browse/browse?sid=1346442495

Supported Sources: Local Media Servers, Playlists, History, Aux-In, Favorites, TuneIn, Pandora, Rhapsody, Deezer, SiriusXM, iHeartRadio, Napster, Tidal, SoundCloud, Amazon Music

### 4.4.4 Browse Source Containers

Command: heos://browse/browse?sid=source_id&cid=container_id&range=start#, end#

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>cid</td>
<td>Container id returned by 'browse' or 'search' command</td>
<td>N/A</td>
</tr>
<tr>
<td>range</td>
<td>Range is start and end record index to return. Range parameter is optional. Omitting range parameter returns all records up to a maximum of either 50 or 100 records per response. The default maximum number of records depend on the service type.</td>
<td></td>
</tr>
<tr>
<td>count</td>
<td>Total number of items available in the container. NOTE: count value of '0' indicates unknown container size. Controllers needs to query until the return payload is empty (returned attribute is 0).</td>
<td></td>
</tr>
<tr>
<td>returned</td>
<td>Number of items returned in current response</td>
<td>N/A</td>
</tr>
<tr>
<td>id (options)</td>
<td>Options available for current browse level</td>
<td>Various options are presented as part of 'Browse Source container' command response. Supported options under each browse menu depends on service type and container type.</td>
</tr>
</tbody>
</table>
The following are valid media types:

- song
- station
- genre
- artist
- album
- container

**Note:** A "yes" for the "container" field as well as the "playable" field implies that the container supports adding all media items to the play queue. Adding all media items of the container to the play queue is performed through "Add containers to queue" command.

**Note:** The option '21 - Playable Container' indicates that the container that is serving the tracks is playable. This option is mainly helpful with the Windows media share. With the Windows media container, HEOS can't determine if a container has playable tracks or not until the container is browsed for its items. When this option appears in browse response, the Controller software could add a virtual item i.e. 'Play all Tracks' along with presenting tracks in the container. When user selects the virtual item, the controller software could then send play command to play the container.

**Note:** Following response provides examples of the various media types. The actual response will depend on the source browsed and the hierarchy supported by that source.

Response:

```json
{
    "heos": {
        "command": "browse/browse",
        "result": "success",
        "message": "sid='source_id&cid='container_id'&range='start,end'&returned=items_in_current_response&count=total_items_available"
    },
    "payload": [
        {
            "container": "yes",
            "playable": "no",
            "type": "artist",
            "name": "artist name",
            "image_url": "artist image url",
            "cid": "container id",
            "mid": "media id"
        },
        {
            "container": "yes",
            "playable": "yes",
            "type": "album",
            "name": "album name",
            "image_url": "album image url",
            "artist": "artist name",
            "cid": "container id",
            "mid": "media id"
        },
        {
            "container": "yes",
            "playable": "yes",
            "type": "song",
            "name": "song name",
            "image_url": "song image url"
        }
    ]
}
```
Example: heos://browse/browse?sid=2&cid=TopAlbums&range=0,100

Supported Sources: Local Media Servers, Playlists, History, Aux-In, TuneIn, Pandora, Rhapsody, Deezer, SiriusXM, iHeartRadio, Napster, Tidal, SoundCloud, Amazon Music

### 4.4.5 Get Source Search Criteria

**Command:** heos://browse/get_search_criteria?sid=source_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>playable</td>
<td>Indicates if Play-All option is supported on searched tracks.</td>
<td>yes or no</td>
</tr>
<tr>
<td>cid</td>
<td>Prefix to search string used while adding entire search results to play queue</td>
<td>Currently supported prefix: SEARCHEDTRACKS-</td>
</tr>
<tr>
<td></td>
<td>Only valid when 'playable' is 'yes'.</td>
<td>Note: Can be extended, avoid hard code</td>
</tr>
<tr>
<td></td>
<td>Example command to play all tracked, searched with string 'earth':</td>
<td></td>
</tr>
<tr>
<td></td>
<td>heos://browse/add_to_queue?pid=&lt;playerid&gt;&amp;sid=2&amp;cid=SEARCHED_TRACKS-earth&amp;aid=1</td>
<td></td>
</tr>
</tbody>
</table>
Note: the following response provides examples of the various search criteria types. The actual response will depend on the source and the search types supported by that source.

Response:

```
{  
  "heos": {  
    "command": "browse/ get_search_criteria ",  
    "result": "success",  
    "message": "sid='source_id "  
  },  
  "payload": [  
    {  
      "name": "Artist",  
      "scid": "search_criteria_id",  
      "wildcard": "yes_or_no",  
    },  
    {  
      "name": "Album",  
      "scid": "search_criteria_id",  
      "wildcard": "yes_or_no",  
    },  
    {  
      "name": "Track",  
      "scid": "search_criteria_id",  
      "wildcard": "yes_or_no",  
      "playable": "yes_or_no",  
      "cid": "Prefix to search string",  
    },  
    {  
      "name": "Station",  
      "scid": "search_criteria_id",  
      "wildcard": "yes_or_no",  
    }  
  ]  
}
```

Example: heos://browse/get_search_criteria?sid=3

Supported Sources: Local Media Servers, TuneIn, Rhapsody, Deezer, SiriusXM, Napster, Tidal, SoundCloud

### 4.4.6 Search

Command: heos://browse/search?sid=source_id&search=search_string&scid=search_criteria&range=start#, end#

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>search</td>
<td>String for search limited to 128 unicode characters and may contain '*' for wildcard if supported by search criteria id</td>
<td>N/A</td>
</tr>
<tr>
<td>scid</td>
<td>Search criteria id returned by 'get_search_criteria' command</td>
<td>artist, album, song, station</td>
</tr>
<tr>
<td>count</td>
<td>Total number of items available in the container. NOTE: count value of '0' indicates unknown container size. Controllers needs to query until the return payload is empty (returned attribute is 0).</td>
<td>0 - unknown, 1+ - valid count</td>
</tr>
<tr>
<td>range</td>
<td>Range is start and end record index to return. Range parameter is optional. Omitting range parameter returns all records up to a maximum of 50/100 records per response. The default maximum number of records depend on the service type.</td>
<td>range starts from 0</td>
</tr>
</tbody>
</table>
Response:

**Note:** the following response provides examples of the various media types. The actual response will depend on the source searched and the results returned for the search string.

```
{
    "heos": {
        "command": "browse/search",
        "result": "success",
        "message": "sid='source_id&scid='search_criteria_id'&range='start#,
        end#&returned=items_in_current_response&count='total_items_available"
    },
    "payload": [
        {
            "container": "yes",
            "playable": "no",
            "type": "artist",
            "name": "artist name",
            "image_url": "artist image url",
            "cid": "container id",
            "mid": "media id"
        },
        {
            "container": "yes",
            "playable": "yes",
            "type": "album",
            "name": "album name",
            "image_url": "album image url",
            "artist": "artist name",
            "cid": "container id",
            "mid": "media id"
        },
        {
            "container": "no",
            "playable": "yes",
            "type": "song",
            "name": "song name",
            "image_url": "album image url",
            "artist": "artist name",
            "album": "album name",
            "mid": "media id"
        },
        {
            "container": "yes",
            "playable": "no",
            "type": "container",
            "name": "container name",
            "image_url": "container image url",
            "cid": "container id",
            "mid": "media id"
        },
        {
            "container": "no",
            "playable": "yes",
            "type": "station",
            "name": "station name",
            "image_url": "station url",
            "mid": "media id"
        }
    ],
    "options": [
        {
            "browse": [
                {
                    "id": 2,
                    "name": "Add Album to Library"
                }
            ]
        }
    ]
}
```
Example: heos://browse/search?sid=2&search="U2"&scid=1

Supported Sources: Local Media Servers, TuneIn, Rhapsody, Deezer, Napster, Tidal, SoundCloud

### 4.4.7 Play Station

Command: heos://browse/play_stream?pid=player_id&sid=source_id&cid=container_id&mid=media_id&name=station_name

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>cid</td>
<td>Container id that is used to browse current container. Ignore if container id doesn't exists as in case of playing station obtained through 'Search' command.</td>
<td>N/A</td>
</tr>
<tr>
<td>mid</td>
<td>Media id returned by 'browse'or 'search' command</td>
<td>N/A</td>
</tr>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>name</td>
<td>Station name returned by 'browse' command.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note: The mid for this command must be a 'station' media type.

Response:

**Note:** this command will cause a Now Playing Change Event to occur if a new stream is played.

```json
{  
    "heos": {  
        "command": "browse/play_stream ",  
        "result": "success",  
        "message": "pid='player_id'&sid='source_id&cid='container_id'&mid='media_id'&name='station_name'"  
    }  
}
```

Example: heos://browse/play_stream?pid=1&sid=2&cid='CID-55'&mid=15376&name=Q95

Supported Sources: History, Favorites, TuneIn, Pandora, Rhapsody, Deezer, SiriusXM, iHeartRadio, Napster, SoundCloud, Amazon Music

### 4.4.8 Play Preset Station

Command: heos://browse/play_preset?pid=player_id&preset=preset_position

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>preset</td>
<td>Station offset in HEOS Favorites</td>
<td>1 and above</td>
</tr>
</tbody>
</table>

Response:

**Note:** this command will cause a Now Playing Change Event to occur if a new stream is played.

```json
{  
    "heos": {  
        "command": "browse/play_preset",  
        "result": "success",  
        "message": "pid='player_id'&preset='preset_number"  
    }  
}
```

Example: heos://browse/play_preset?pid=1&preset=2
Supported Sources: HEOS Favorites

4.4.9 Play Input source

Command to play input source on the same speaker:

heos://browse/play_input?pid=player_id&input=input_name

Command to play input source on another speaker:

heos://browse/play_input?pid=destination_player_id&spid=source_player_id&input=input_name

OBSOLETE command that requires sid:

heos://browse/play_stream?pid=player_id&sid=source_id&mid=media_id

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>pid</td>
<td>player id of the selected speaker (destination HEOS speaker)</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td></td>
</tr>
<tr>
<td>mid</td>
<td>media id returned by 'browse' command</td>
<td>N/A</td>
</tr>
<tr>
<td>spid</td>
<td>player id of the HEOS device which is acting as the source</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td></td>
</tr>
<tr>
<td>input</td>
<td>input source name</td>
<td>&quot;inputs/aux_in_1&quot; &quot;inputs/aux_in_2&quot; &quot;inputs/aux_in_3&quot; &quot;inputs/aux_in_4&quot; &quot;inputs/aux_single&quot; &quot;inputs/aux1&quot; &quot;inputs/aux2&quot; &quot;inputs/aux3&quot; &quot;inputs/aux4&quot; &quot;inputs/aux5&quot; &quot;inputs/aux6&quot; &quot;inputs/aux7&quot; &quot;inputs/line_in_1&quot; &quot;inputs/line_in_2&quot; &quot;inputs/line_in_3&quot; &quot;inputs/line_in_4&quot; &quot;inputs/coax_in_1&quot; &quot;inputs/coax_in_2&quot; &quot;inputs/optical_in_1&quot; &quot;inputs/optical_in_2&quot; &quot;inputs/hdmi_in_1&quot; &quot;inputs/hdmi_in_2&quot; &quot;inputs/hdmi_in_3&quot; &quot;inputs/hdmi_in_4&quot; &quot;inputs/hdmi_arc_1&quot; &quot;inputs/cable_sat&quot; &quot;inputs/dvd&quot; &quot;inputs/blu-ray&quot; &quot;inputs/game&quot; &quot;inputs/mediaplayer&quot; &quot;inputs/cd&quot; &quot;inputs/tuner&quot; &quot;inputs/hdradio&quot; &quot;inputs/tv_audio&quot;</td>
</tr>
</tbody>
</table>

Note: Validity of Inputs depends on the type of source HEOS device
Response for command "heos://browse/play_input?pid=player_id&input=input_name" :

Note: this command will cause a Now Playing Change Event to occur if an aux in stream is played.

```
{
  "heos": {
    "command": "browse/play_input",
    "result": "success",
    "message": "pid=player_id&input=input_name"
  }
}
```

Limitations for the system when used multi devices.

- Distribution External Input to other players is limited to one player or one group.
  You can not play External Input that has been selected already.
  Also, When it playing external input on itself, can not be distributed to other players.

Examples:

heos://browse/play_input?pid=1234&input=inputs/aux_in_1
heos://browse/play_input?pid=1234&spid=9876&input=inputs/aux_in_1
heos://browse/play_stream?pid=1&sid=1441320818&mid=inputs/aux_in_1

4.4.10 Play URL

Command: heos://browse/play_stream?pid=player_id&url=url_path

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
<tr>
<td>url</td>
<td>Absolute path to a playable stream</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

Note: The attribute value pair "url=url_path" should be the last attribute value pair in the play_stream command.
This is required to handle url_path with special characters and command delimiters.
This command will cause a Now Playing Change Event to occur if url is played.

```
{
  "heos": {
    "command": "browse/play_stream",
    "result": "success",
    "message": "pid=player_id&url='path to stream"
  }
}
```


4.4.11 Add Container to Queue with Options

Command: heos://browse/add_to_queue?pid=player_id&sid=source_id&cid=container_id&aid=add_criteria

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>cid</td>
<td>Container id returned by 'browse' or 'search' command</td>
<td>N/A</td>
</tr>
<tr>
<td>aid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Add criteria id as defined by enumerations ->

1 – play now
2 – play next
3 – add to end
4 – replace and play

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>Player id returned by 'get_players' or 'get_groups' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note: The cid for this command must be a ‘playable’ container type.

Response:

Note: this command will cause a Now Playing Change Event to occur if a new song is played.

```
{  
    "heos": {  
        "command": "browse/add_to_queue",  
        "result": "success",  
        "message": "pid='player_id'&sid='source_id'&cid='container_id'&mid='media_id'&aid='add_criteria'"  
    }  
}
```

Example: heos://browse/add_to_queue?pid=1&sid=8&cid=Artists/All&mid=9&aid=1

Supported Sources: Playable containers from Local Media Servers, Playlists, History, Rhapsody, Deezer, iHeartRadio, Napster, Tidal, SoundCloud. Also searched tracks as described in get_search_criteria command.

**4.4.12 Add Track to Queue with Options**

Command: heos://browse/add_to_queue?pid=player_id&sid=source_id&cid=container_id&mid=media_id&aid=add-criteria

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>cid</td>
<td>Container id that is used to 'browse' or 'search' current container</td>
<td>N/A</td>
</tr>
<tr>
<td>mid</td>
<td>Media id returned by 'browse' or 'search' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| aid       | Add criteria id as defined by enumerations -> | 1 – play now
2 – play next
3 – add to end
4 – replace and play |
| pid       | Player id returned by 'get_players' or 'get_groups' command | N/A         |

Note: The mid for this command must be a ‘track’ media type.

Response:

Note: this command will cause a Now Playing Change Event to occur if a new song is played.

```
{  
    "heos": {  
        "command": "browse/add_to_queue",  
        "result": "success",  
        "message": "pid='player_id'&sid='source_id'&cid='container_id'&mid='media_id'&aid='add_criteria'"  
    }  
}
```

Example: heos://browse/add_to_queue?pid=1&sid=8&cid=Artist/All&mid=9&aid=2

Supported Sources: Local Media Servers, Playlists, History, Rhapsody Tracks, Deezer Tracks, iHeartRadio Tracks, Napster, Tidal, SoundCloud, Amazon Music. Please note Amazon Music tracks are played without adding to HEOS queue.

**4.4.13 Get HEOS Playlists**

Refer to Browse Sources and Browse Source Containers
### 4.4.14 Rename HEOS Playlist

Command: `heos://browse/rename_playlist?sid=source_id&cid=container_id&name=playlist_name`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command; select HEOS source to get HEOS playlists.</td>
<td>N/A</td>
</tr>
<tr>
<td>cid</td>
<td>Container id returned in 'Get HEOS Playlists' command</td>
<td>N/A</td>
</tr>
<tr>
<td>name</td>
<td>String for new playlist name limited to 128 unicode characters</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

```
{  
  "heos": {
    "command": "browse/rename_playlist ",
    "result": 'success',
    "message": "sid='source_id'&cid='container_id'&name='playlist_name'"
  }
}
```

Example: `heos://browse/rename_playlist?sid=11&cid=234&name=new name`

### 4.4.15 Delete HEOS Playlist

Command: `heos://browse/delete_playlist?sid=source_id&cid=container_id`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command; select HEOS source to get HEOS playlists.</td>
<td>N/A</td>
</tr>
<tr>
<td>cid</td>
<td>Container id returned in 'Get HEOS Playlists' command</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Response:

```
{  
  "heos": {
    "command": "browse/delete_playlist ",
    "result": 'success',
    "message": "sid='source_id'&cid='container_id'"
  }
}
```

Example: `heos://browse/delete_playlist?sid=11&cid=234`

### 4.4.16 Get HEOS History

Refer to Browse Sources and Browse Source Containers

### 4.4.17 Retrieve Album Metadata

Rhapsody and Napster services don't provide album art url while browsing for tracks. Controllers can use this command to retrieve album art url while browsing for tracks.

Retrieve image url associated with a given album id. This command facilitates controllers to retrieve and update their UI with cover art, if image_url in `browse/search/get_queue/get_now_playing_media` command response is blank.

Command: `heos://browse/retrieve_metadata?sid=source_id&cid=album_id`
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command; select HEOS source to get HEOS playlists.</td>
<td>Currently supported media sources are Rhapsody/Napster</td>
</tr>
<tr>
<td>cid</td>
<td>Container id returned by 'browse' command or 'get_now_playing_media' command</td>
<td>Rhapsody/Napster album ids</td>
</tr>
</tbody>
</table>

Note: Supported music service is Rhapsody and Napster

Response:
```
{
  "heos": {
    "command": "browse/retrieve_metadata",
    "result": "success",
    "message": "sid=2&cid=album_id&returned=items_in_current_response&count=total_items_available"
  },
  "payload": {
    "album_id": "album_id",
    "images": [
      {
        "image_url": "URL to image file",
        "width": current image width
      },
      {
        "image_url": "URL to image file",
        "width": current image width
      }
    ]
  }
}
```

Example: `heos://browse/retrieve_metadata?sid=2&cid=Alb.184664171`

### 4.4.18 Get Service Options for now playing screen - OBSOLETE

Obsolete - Now get_now_playing_media command will include supported option for currently playing media.

Command: `heos://browse/get_service_options?sid=source_id`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>Source id returned by 'get_music_sources' command</td>
<td>N/A</td>
</tr>
<tr>
<td>id (options)</td>
<td>Options available on now playing screen</td>
<td>Following options are currently supported for 'Get Service options for now playing screen': 11 - Thumbs Up 12 - Thumbs Down</td>
</tr>
</tbody>
</table>

**Note:** This command returns service options that are only available on 'now playing' screen. Please refer to 'Browse Source' and 'Browse Source Containers' for service options available on various browse levels.

**Note:** the following response provides examples of the various service options. The actual response will depend on the service options available for a given source type.
Response:

```
{
  "heos": {
    "command": "browse/get_service_options",
    "result": "success",
    "message": ""
  },
  "payload": {
    "play": [
      {
        "id": 11,
        "name": "Thumbs Up"
      },
      {
        "id": 12,
        "name": "Thumbs Down"
      }
    ]
  }
}
```

Example: heos://browse/get_service_options?sid=5

### 4.4.19 Set service option

Set service option is a generic command used to select any of the supported service options provided through 'Get Service Options for now playing screen', 'Browse Sources' and 'Browse Source Containers' command response.

Following service options are currently supported:

<table>
<thead>
<tr>
<th>Option id</th>
<th>Example Command</th>
<th>Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - Add Track to Library</td>
<td>heos://browse/set_service_option?sid=2&amp;option=1&amp;mid=Tra.174684187</td>
<td>mid - track id obtained through 'browse source containers' command</td>
</tr>
<tr>
<td>Supported Services: Napster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - Add Album to Library</td>
<td>heos://browse/set_service_option?sid=2&amp;option=2&amp;cid=Alb.174684186</td>
<td>cid - album id obtained through 'browse source containers' command</td>
</tr>
<tr>
<td>Supported Services: Napster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - Add Station to Library</td>
<td>heos://browse/set_service_option?sid=2&amp;option=3&amp;mid=sas.6513639</td>
<td>mid - station id obtained through 'browse source containers' command</td>
</tr>
<tr>
<td>Supported Services: Napster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 - Add Playlist to Library</td>
<td>heos://browse/set_service_option?sid=2&amp;option=4&amp;cid=LIBPLAYLIST-pp.1755731498&amp;name=Lupe Fiasco</td>
<td>cid - playlist id obtained through 'browse source containers' command name - name of the playlist obtained through 'browse source container' command</td>
</tr>
<tr>
<td>Supported Services: Napster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 - Remove Track from Library</td>
<td>heos://browse/set_service_option?sid=2&amp;option=5&amp;mid=Tra.174684187</td>
<td>mid - track id obtained through 'browse source containers' command</td>
</tr>
<tr>
<td>Supported Services: Napster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 - Remove Album from Library</td>
<td>heos://browse/set_service_option?sid=2&amp;option=6&amp;cid=LIBALBUM-Alb.174684186</td>
<td>cid - album id obtained through 'browse source containers' command</td>
</tr>
<tr>
<td>Supported Services: Napster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 - Remove Station from Library</td>
<td>heos://browse/set_service_option?sid=2&amp;option=7&amp;mid=sas.6513639</td>
<td>mid - station id obtained through 'browse source containers' command</td>
</tr>
<tr>
<td>Supported Services: Napster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 - Remove Playlist from Library</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Supported Services: Napster

**Supported Services**

- Napster

**Supported Services provide this option in Get Now Playing Media command response**

```
heos://browse/set_service_option?sid=2&option=1&mid=sas.186017722
```

**cid** - playlist id obtained through 'browse source containers' command

**11 - Thumbs Up**

**Supported Services provide this option in Get Now Playing Media command response**

```
heos://browse/set_service_option?sid=1&option=11&pid=-409995282
```

**pid** - player id obtained through 'get_players' command

### 12 - Thumbs Down

**Supported Services provide this option in Get Now Playing Media command response**

```
heos://browse/set_service_option?sid=1&option=12&pid=-409995282
```

**pid** - player id obtained through 'get_players' command

### 13 - Create New Station by Artists

**Supported Services: Pandora, iHeartRadio**

```
heos://browse/set_service_option?sid=1&option=13&name=Love&scid=1
```

name - search string for creating new station

**Note:** This command returns station ids. Controllers need to use 'play station' command to play a station.

**Note:** Range parameter is optional to limit results

```
heos://browse/set_service_option?sid=1&option=13&name=Love&range=0,10
```

### 13 - Create New Station by Shows

**Supported Services: iHeartRadio**

```
heos://browse/set_service_option?sid=7&option=13&name=Love&scid=5
```

name - search string for creating new station through show

**Note:** This command returns station ids. Controllers need to use 'play station' command to play a station.

```
heos://browse/set_service_option?sid=7&option=13&name=Love&range=0,10
```

### 13 - Create New Station by Tracks

**Supported Services: iHeartRadio**

```
heos://browse/set_service_option?sid=7&option=13&name=Love&scid=3
```

name - search string for creating new station through track

**Note:** This command returns station ids. Controllers need to use 'play station' command to play a station.

```
heos://browse/set_service_option?sid=7&option=13&name=Love&range=0,10
```

### 19 - Add station to HEOS Favorites

**Following command is used on now-playing-screen to add currently playing station to HEOS Favorites**

```
heos://browse/set_service_option?option=19&pid=-409995282
```

**pid** - player id obtained through 'get_players' command

**Following command is used on browse screen to add a station to HEOS Favorites**

```
heos://browse/set_service_option?option=19&mid=sas.6513639&name=Folk Radio
```

**mid** - station id obtained through 'browse source' command on Favorites

**name** - station name obtained through 'browse' command

### 20 - Remove from HEOS Favorites

```
heos://browse/set_service_option?option=20&mid=sas.6513639
```

**mid** - station id obtained through 'browse source' command on Favorites

**Response:**

```json
{
  "heos": {
    "command": "browse/set_service_option",
    "result": "success",
    "message": "sid=source_id&option=option_id&mid=media_id"
  }
}
```

**Example:** heos://browse/set_service_option?sid=2&option=1&mid=Tra.174684187

**Note:** Option 13 (Create New Station) supports optional range queries.
5. Change Events (Unsolicited Responses)

5.1 Sources Changed
Response:
{
    "heos": {
        "command": "event/sources_changed",
    }
}

5.2 Players Changed
Response:
{
    "heos": {
        "command": "event/players_changed",
    }
}

5.3 Group Changed
Response:
{
    "heos": {
        "command": "event/groups_changed",
    }
}

5.4 Player State Changed
Response:
{
    "heos": {
        "command": "event/player_state_changed",
        "message": "pid='player_id'&state='play_state'"
    }
}

5.5 Player Now Playing Changed
Response:
{
    "heos": {
        "command": "event/player_now_playing_changed",
        "message": "pid='player_id'"
    }
}

5.6 Player Now Playing Progress
Response:
{
    "heos": {
        "command": "event/player_now_playing_progress",
        "message": "pid=player_id&cur_pos=position_ms&duration=duration_ms"
    }
}
5.7 Player Playback Error

Response:

```json
{  
  "heos": {
    "command": "event/player_playback_error",
    "message": "pid=player_id&error=Could Not Download"
  }
}
```

Note: error string represents error type. Controller can directly display the error string to the user.

5.8 Player Queue Changed

Response:

```json
{  
  "heos": {
    "command": "event/player_queue_changed",
    "message": "pid='player_id'"
  }
}
```

5.9 Player Volume Changed

Response:

```json
{  
  "heos": {
    "command": "event/player_volume_changed",
    "message": "pid='player_id'&level='vol_level'&mute='on_or_off'"
  }
}
```

5.10 Player Repeat Mode Changed

Response:

```json
{  
  "heos": {
    "command": "event/repeat_mode_changed",
    "message": "pid='player_id'&repeat='on_all_or_on_one_or_off'"
  }
}
```

5.11 Player Shuffle Mode Changed

Response:

```json
{  
  "heos": {
    "command": "event/shuffle_mode_changed",
    "message": "pid='player_id'&shuffle='on_or_off'"
  }
}
```

5.12 Group Volume Changed

Response:

```json
{  
  "heos": {
    "command": "event/group_volume_changed",
    "message": "gid='group_id'&level='vol_level'&mute='on_or_off'"
  }
}
```
5.13 User Changed

Response:
{
    "heos": {
        "command": "event/user_changed",
        "message": "signed_out" or "signed_in&un=<current user name>"
    }
}

6.0 Error Codes

6.1 General Error Response

Response:
{
    "heos": {
        "command": "command_group'/command",
        "result": "fail",
        "message": "eid="error_id&text=error text&command_arguments"
    }
}

6.2 Error Code description

<table>
<thead>
<tr>
<th>Description</th>
<th>Code</th>
<th>Text Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unrecognized Command</td>
<td>1</td>
<td>Command not recognized.</td>
</tr>
<tr>
<td>Invalid ID</td>
<td>2</td>
<td>ID not valid</td>
</tr>
<tr>
<td>Wrong Number of Command Arguments</td>
<td>3</td>
<td>Command arguments not correct.</td>
</tr>
<tr>
<td>Requested data not available</td>
<td>4</td>
<td>Requested data not available.</td>
</tr>
<tr>
<td>Resource currently not available</td>
<td>5</td>
<td>Resource currently not available.</td>
</tr>
<tr>
<td>Invalid Credentials</td>
<td>6</td>
<td>Invalid Credentials.</td>
</tr>
<tr>
<td>Command Could Not Be Executed</td>
<td>7</td>
<td>Command not executed.</td>
</tr>
<tr>
<td>User not logged in</td>
<td>8</td>
<td>User not logged in.</td>
</tr>
<tr>
<td>Parameter out of range</td>
<td>9</td>
<td>Out of range</td>
</tr>
<tr>
<td>User not found</td>
<td>10</td>
<td>User not found</td>
</tr>
<tr>
<td>Internal Error</td>
<td>11</td>
<td>System Internal Error</td>
</tr>
<tr>
<td>System Error</td>
<td>12</td>
<td>System error&amp;syserrno=XXX</td>
</tr>
</tbody>
</table>

Popular system errors are:

-9: Indicates that the action was sent to the remote service, but the remote service returned an error response
-1061: The user has not registered this service
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1063</td>
<td>The user is not logged in</td>
</tr>
<tr>
<td>-1056</td>
<td>User not found</td>
</tr>
<tr>
<td>-1201</td>
<td>General content services authentication error</td>
</tr>
<tr>
<td>-1232</td>
<td>Content services user authorization error</td>
</tr>
<tr>
<td>-1239</td>
<td>User account related parameters are invalid</td>
</tr>
<tr>
<td>Processing Previous Command</td>
<td>13</td>
</tr>
<tr>
<td>Media can't be played</td>
<td>14</td>
</tr>
<tr>
<td>Option no supported</td>
<td>15</td>
</tr>
<tr>
<td>Too many commands in message queue to process</td>
<td>16</td>
</tr>
<tr>
<td>Reached skip limit</td>
<td>17</td>
</tr>
</tbody>
</table>