Logitech Media Server CLI

LMS/CLI Version 7.7.5

Ein Tutorial zur Steuerung des LMS mittels Loxone findest du hier.

Command Line Reference (CLI)

Introduction

the Logitech Media Server provides a command-line interface to the players via TCP/IP. After starting the server, commands and queries may be sent by connecting to a specific TCP/IP port. The server will reply echoing the request (for commands) or by returning the requested data (for queries). By default, the server will listen for connections on TCP/IP port 9090. This format is designed for ease of integration into AMX, Crestron and other automation systems.

The end of line separator is line feed (<LF> ASCII decimal 10, hexadecimal 0x0A). The server accepts LF, CR or 0x00 (or any combination thereof) as end of line, and replies with whatever was used for the command. For strings, Logitech Media Server uses the UTF-8 character-set encoding.

To use the command line interface interactively, use the telnet command from your system's command prompt: telnet localhost 9090 and when it connects, you can start typing commands. Beware that the server expects parameters to be encoded using percent-style escaping (see below); " and \ are not supported as in shell-like environments.

For debugging purposes, CLI formatted commands can be sent using standard in and out. This support is only available on Unix platforms (MacOS X included), and must be enabled by launching the server with the "--stdio" command line option.

Changes starting from Squeezebox Server 7.7

- Extended "rescan" with optional singledir parameter

Changes starting from Squeezebox Server 7.6

- Added tag 'M' for songinfo to return track musicmagic_mixable value.
- Added tag 'c' for songinfo to return the track's coverid value, used for artwork URLs. If you were using the track's ID directly in artwork URLs, you should switch to using the coverid value.
- Added "pragma" command for changing SQLite behavior.
- Added artist_id tagged parameter to artists.
- Added album_id tagged parameter to albums.
- Added sort order 'albumtrack' for titles to return the tracks in the order album-title, track-number.
- Added tag 'X' for albums to return the album's replay-gain value.
- Added tag 'S' for albums to return the album's artist_id.
- Added 'play_index' tagged parameter to playlistcontrol.
- In extended queries, return all possible items if the <itemsPerResponse> parameter is omitted.
- Add the 'u' tag to the "musicfolder" query
- Added track_id tagged parameter to titles.
- Added genre_id tagged parameter to genres.
- Added year and hasAlbums tagged parameters to years.
- Added return_top tagged parameter to musicfolder.
- Removed the charset tagged parameter from all commands that supported it.
- Added the 'Z' tag to albums, artists and genres queries. It generates a indexList array or arrays in the result.

Changes starting from Squeezebox Server 7.5.1

- Added "playlist preview" command.

- Extended "playlist resume" with optional noplay and wipePlaylist tagged params.
- Extended "playlist save" with optional silent tagged param, which will suppress any showBriefly popup on squeezeplay players.

Changes starting from Squeezebox Server 7.4

- The music_services command is now apps.
- Added "pause" & "stop" subtypes to "playlist" notifications.
- Added "setsncredentials" command.
- Added "logging" command.

Changes starting from Squeezebox Server 7.3

- Added 'icon' tag to radios and music_services response.
- Added "syncgroups" query to get a list of sync groups and their members.
- Added "favorites exists" query to check whether an item exists in favorites.

Changes starting from Squeezebox Server 7.2

The following syntax changes apply to the CLI in Squeezebox Server 7.2. These changes may impact CLI clients.

- The "alarm" and "alarms" command and query have been modified to match the improved alarm functionality. Most important change is the switch back to Sun=0 to Sat=6 days, instead of 0-7, where 0 meant "everyday" and Sun=7.

The following changes or new commands & queries are available starting with Squeezebox Server 7.2. These changes should not have any impact on existing clients:

- New notification mechanism for alarms: "alarm <sound|end|snooze|snooze_end> <id>".
- New query to get localized strings: "getstring <STRINGTOKEN1[,STRINGTOKEN2...]>".

Changes starting from Squeezebox Server 7.0

The following syntax changes apply to the CLI in Squeezebox Server 7.0. These changes may impact CLI clients.

- Completely modified the Favorites plugin support. It is now based on XMLBrowser, like most of the internet radios, and documented in the Plugins section of this document.
- Completely modified live365 support. It is now based on XMLBrowser, like all internet radios.
- Deprecated the tag "year_id" in favour of "year" in the "playlistcontrol" command.
- Changed how the "status" query behaves if the player one is subscribed to disappears.
- Updated the "status" query to return the time stamp of the last change to the current player playlist.
- Updated the "playlist newsong" notification to return the song title and playlist index.
- Updated the "pref" and "playerpref" commands to support the namespaces for preferences.
- Changed tags in queries "status", "titles", "playlists tracks" and "songinfo" for artist(s)/contributor(s) (band, composer, conductor, trackartist, etc) and genre(s). Multiple items may now be returned.
- New notification mechanism for preference changes: "prefset".

The following changes or new commands & queries are available starting with Squeezebox Server 7.0. These changes should not have any impact on existing clients:

- Added the "years" query, to enable Browse by Year functionality.
- Added the "musicfolder" query, to enable Browse Music Folder functionality.
- Added the "readdirectory" query, to browse file systems from the server's point of view (local & network shares)
- Added the "rescanprogress" query, to report details on scanning progress.
- Added the "abortscan" command, to stop a running scan.
- Added the "serverstatus" query, to return compound server status in a single query.
- Added the "playlists new" command to create a stored playlist.
- Added the "prefs" tag to the "players" query, to enable returning the given preference values along with each player.
- Added the "canpoweroff", "isplayer" and "uuid" tags to the "players", "player" and "serverstatus" queries. to enable returning the given preference values along with each player.
- Added the "pref validate" and "playerpref validate" queries to validate preferences without setting them.
- Added a tag to return the artist from the "albums" query.
- Added a tag to return if the item is audio from the "XML based" queries.

- Added a tag to allow sorting the results of the "radios" query.
- Added the "name" query/command to get/change the player name.
- Added the "irenable" query/command to enable/disable IR processing for a player.
- Added the "displaystatus" query which allows subscription to display update messages.
- Updated queries "status", "titles", "playlists tracks" and "songinfo" to support a new tag J to return the artwork_track_id (as returned by the "albums"). This enables clients to cache one image for all songs of the same album.
- Updated queries "status", "titles", "playlists tracks" and "songinfo" to support new tags for previously missing information like sample rate/size, rating, etc.
- Order of tracks passed to "playlistcontrol" command is maintained.
- Command "playlistcontrol" now accepts also a folder_id as returned by the new "musicfolder" query.
- Slightly reorganised this document to introduce a "Compound queries" section to document queries "serverstatus" and "status".

Command format

General command and query format

The format of the commands, queries and server replies is as follows:

> [<playerid>] <p0> <p1> ... <pN> <LF>

where:

> <playerid> is the unique identifier for the player, usually (but not guaranteed to be) the MAC address of the player. Some commands are global to the server and do not require a <playerid>. For commands requiring it, a random player will be selected by the server if the <playerid> is omitted, and returned in the server reply. <playerid> may be obtained by using the "player id" or "players" queries.
>
> <p0> through <pN> are positional parameters. Pass a "?" to obtain a value for that parameter in the server response (i.e. send a query). Details of the parameters vary for each command as described below.

Each parameter needs to be encoded using percent-style escaping, the same method as is used in URLs; for example, "The Clash?" would be encoded as "The%20Clash%3F". This also applies to <playerid>. In the examples below, the escaping is not show for readability (except %20 for space).

Extended query format

A few extended queries are defined, to regroup multiple queries and allow browsing the server database. These queries return multiple items. Overall however, their format is compliant with the general command format above:

> [<playerid>] <command> <start> <itemsPerResponse> <p3> ... <pN> <LF>

where:

> <playerid> is the unique identifier for the player, as above.
>
> <command> is the query name.
>
> <start> and <itemsPerResponse> are positional parameters that control the response chunking. <start> is a zero-based index of the first item to return, and <itemsPerResponse> is the number of items to return, if possible.
>
> <p3> through <pN> are tagged parameters. Tags consist of a name followed by ":". For example, "artist:Abba". The tag and the ":" are URL escaped with the field value. Tag names cannot contain ":" but the data can.

to which the server replies:

> [<playerid>] <command> <start> <itemsPerResponse> <p3> ... <pN> <pN+1> ... <pM> <LF>

where:

> The entire query is repeated.
>
> <pN+1> through <pM> is the tagged returned data. A special tag value is defined in each command to separate the multiple returned items. Data is only returned when applicable, that is, all possible tags are not always returned.
>
> If the <itemsPerResponse> positional parameter and all tagged parameters are omitted then all possible items are returned.

Example: The "players" command returns data on all players known by the server. It is a shortcut call compared to the general CLI API "player count ?" followed by a number of calls to get the players name, ID, etc. The command must be called with the chunking parameters. For example, the following returns information on the first 10 players known by the server (if so many exist), starting from

the first one:

Request: "players 0 2<LF>"

Response: "players 0 2 count:2 playerindex:0 playerid:a5:41:d2:cd:cd:05 ip:127.0.0.1:60488 name:127.0.0.1 model:softsqueeze displaytype:graphic-280x16 connected:1 playerindex:1 playerid:00:04:20:02:00:c8 ip:192.168.1.22:3483 name:Movy model:slimp3 displaytype:noritake-katakana connected:1<LF>"

Extended command format

Extended commands are commands that reuse the general principle of tagged parameters as introduced by extended queries:

[<playerid>] <command> <p1> ... <pN> <LF>

where:

<playerid> is the unique identifier for the player, as above.

<command> is the command name.

<p1> through <pN> are tagged parameters as defined above.

The server performs the command and returns:

[<playerid>] <command> <p1> ... <pN> <pN+1> ... <pM> <LF>

where:

The entire query is repeated.

<pN+1> through <pM> is the tagged returned data. See the command description for definitions. In general commands do return some information about the command performed.

Notes

- The Security settings of the server preferences apply to CLI connections when they are established. A change in security settings do not affect established connections. The connection is only accepted from allowed hosts. If password protection is enabled, the "login" command must be the first command sent after the connection. Any error in the user and/or password, or using any other command as the first one, results in the server disconnecting.
- All Logitech Media Server preferences apply to the CLI data. For examples, the preference about composers appearing in the artists list applies to the data returned by the "artists" query.
- Commands that use paths to songs or playlists (<item> parameters below) can use relative paths from the root of the Music Library folder to specify songs. For example, if the Music Library is specified as "D:\mymusic" and you'd like to refer to a song in that folder named "foo.mp3" you can specify just "foo.mp3" in the command parameter. Likewise, to refer to items in the Saved Playlist folder, you can use a prefix of "__playlists/" before the path. For example, to refer to the saved playlist "bar.m3u" in the Saved Playlists folder, you can specify a path of "__playlists/bar.m3u".
- The HTTP server can return cover art for songs using the track ID as returned by the CLI functions. If no cover art exists for the given song, the server returns special "no artwork" image. Please refer to the Artwork Setup documentation for more details on artwork management in Logitech Media Server. Use the following URL:
    - http://<server>:<port>/music/<track_id>/cover.jpg
- where:
    - <server> is the ip address or name of the server.
    - <port> is the HTTP port of the server (not the same as the CLI port).
    - <track_id> is the track ID as returned by the CLI functions.
- In addition, there is a shortcut URL to return the artwork of the currently playing song for a player:
    - http://<server>:<port>/music/current/cover.jpg?player=<playerid>
- where:
    - <server> is the ip address or name of the server.
    - <port> is the HTTP port of the server (not the same as the CLI port).
    - <playerid> is the unique identifier for the player, as above. If omitted, the server will use a random player.
- For commands using positional parameters, any extra parameters (after all required ones) will be returned. For commands using tagged parameters, parameters using unknown tags will be returned as well. This allows the client to add to commands and queries some context information. For example:
    - Request: "04:20:00:12:23:45 mixer bass ? context<LF>"
    - Response: "04:20:00:12:23:45 mixer bass 98 context<LF>"
    - Request: "players 0 2 context:1<LF>"
    - Response: "players 0 2 context:1 count:2 id:00:04:20:02:00:c8 ...(same as above)"
- All paths returned as URLs, for example the ones returned by the query "songinfo" are double URL escaped. To get a

useable path (that you can use with your file system), you will need to unescape the field twice. Also note the URLs are not translated or re-encoded: they use the encoding of the underlying filesystem API, typically (but not necessarily) the current locale.

- **Transporter Digital Inputs**
- Transporter Digital Inputs are handled as remote streams, with a URL starting with source: followed by "aes-ebu", "bnc-spdif", "toslink" or "rca-spdif".
- To set Transporter to the TOSLINK input, use "<playerid> playlist play source:toslink<LF>".
- When set to a digital input, Transporter reports the URL scheme above to the various path, url or status queries.

Supported Commands

The available commands and queries are listed below, grouped by their scope:

- General: general management of the Command Line Interface and of the server.
- Players: management of players.
- Database: mangement of the music database.
- Playlist: management of the playlist of each player.
- Coumpound queries: queries to get most of the information about the server or a player in one convenient query, that can be updated by the server automatically.
- Notifications: internal server commands echoed to the CLI by using the "listen" or "subscribe" commands.
- Alarms: management of alarms.
- Plugins: commands and queries proposed by various server plugins. These are only available if the plugin is enabled in the server configuration. The query "can" can be used to determine if a command or query is available.
- Deprecated: commands which are still available but not for much longer...

General commands and queries

- login
- can
- version
- listen
- subscribe
- pref
- logging
- getstring
- setsncredentials
- debug
- exit
- shutdown

login <user> <password>

The "login" command allows the caller to authenticate itself on the server, as defined in the Security pane of the server preferences. Like any other command, the user and password must be escaped. If successful, the server replaces the password with 6 star characters. If unsuccessful, the server disconnects. If security is off this command is always successful.

Examples:

Request: "login user correctpassword<LF>"

Response: "login user ******<LF>"

Request: "login user wrongpassword<LF>"

Response: (Connection terminated)

can <request terms> ?

The "can" query allows the caller to determine if the command or query indicated by <request terms> is available.

Examples:

Request: "can info total genres ?<LF>"

Response: "can info total genres 1<LF>"

Request: "can smurf ?<LF>"

Response: "can smurf 0<LF>"

version ?

The "version" query returns version number of the server.

Examples:

Request: "version ?<LF>"

Response: "version 6.5<LF>"

listen <0|1|?>

The "listen" command enables to receive asynchronously internal server commands (notifications) on the CLI connection. Notifications concern all activity in the server, not just the activity triggered by the command-line. Use 0 to clear, 1 to set, ? to query, and no parameter to toggle the listen state of the command-line connection.

If only certain notifications are of interrest, consider using the "subscribe" command below. The "listen" command shares some of its internal plumbing with "subscribe" so using "subscribe xxx" changes the list of echoed notifications from nothing or everything to only xxx.

Please consult section Notifications for a list of possible notifications.

Examples:

Request: "listen 1<LF>"

Response: "listen 1<LF>"

"04:20:00:12:23:45 mixer volume 25<LF>"

"04:20:00:12:23:45 pause<LF>"

...

subscribe <comma_separated_notification_list>

The "subscribe" command is similar to "listen" but echoes only a subset of the notifications, indicated by a comma separated list. If no list is provided, the notifications are turned off. This command shares some of its internal plumbing with "listen" so using "listen 0" or "listen 1" changes the list of echoed notifications (to nothing and everything, respectively).

Please consult section Notifications for a list of possible notifications.

Examples:

Request: "subscribe mixer,pause<LF>"

Response: "subscribe mixer,pause<LF>"

"04:20:00:12:23:45 mixer volume 25<LF>"

"04:20:00:12:23:45 pause<LF>"

...

pref <prefname|namespace:prefname> <prefvalue|?>

The "pref" command allows the caller to set and query the server's internal preference values. The following affect the behaviour of CLI queries and commands:

- **authorize**: Security enabled or not. If enabled, usage of the "login" command is required.
- **groupdiscs**: handling of multiple disc sets. Affects the "albums" query.
- **variousArtistAutoIdentification**: compilation artists are listed as "Various Artists". Affects the "artists" query.

- **splitList**: delimiter for multiple items in tags. Affects all the queries returning genres, mainly "genres".
- **composerInArtists**, **conductorInArtists**, **bandInArtists**: contributors considered artists. Affects the "info total artists ?" query.

If you want to query/set a preference from an other namespace than "server" (eg. a plugin), you'll have to prepend the desired namespace to the prefname.

Examples:

Request: "pref audiodir ?<LF>"

Response: "pref audiodir %2fUsers%2fdean%2fDesktop%2ftest%20music<LF>"

Request: "pref plugin.rescan:time ?<LF>"

Response: "pref plugin.rescan:time 32400<LF>"

Request: "pref playlistdir %2fUsers%2fdean%2fplaylists<LF>"

Response: "pref playlistdir %2fUsers%2fdean%2fplaylists<LF>"

logging <group:logging group> [<persist:1>]

The "logging" command allows setting some logging levels. Today you can only set one of the following logging groups: server, radio, transcoding, scanner. The optional persist parameter defines whether the change should be persistent or not.

Examples:

Request: "logging group:scanner<LF>"

Response: "logging group:scanner<LF>"

pref validate <prefname|namespace:prefname> <prefvalue>

The "pref validate" command allows the caller to validate a server's internal preference value without setting it.

If you want to validate a preference from an other namespace than "server" (eg. a plugin), you'll have to prepend the desired namespace to the prefname.

Examples:

Request: "pref validate bufferSecs 10"

Response: "pref validate bufferSecs valid:1"

Request: "pref validate audiodir %2fsome%2fincorrect%2ffilepath"

Response: "pref validate audiodir valid:0"

getstring <STRINGTOKEN1[,STRINGTOKEN2...]>

The "getstring" command allows the caller to query one or several localized strings. String tokens can be passed as a single, concatenated value.

Examples:

Request: "getstring HOME <LF>"

Response: "getstring HOME:Startseite <LF>"

Request: "getstring SETTINGS,SCREENSAVERS,HOME <LF>"

Response: "getstring SETTINGS:Einstellungen SCREENSAVERS:Bildschirmschoner HOME:Startseite <LF>"

setsncredentials <username> <password> [<sync:0|1>]

The "setsncredentials" command allows the caller to change the credentials used to acces mysqueezebox.com resources.

Examples:

> Request: "setsncredentials me@mysqueezebox.com p@s5w0rd <LF>"
>
> Response: "setsncredentials me@mysqueezebox.com p@s5w0rd validated:1 warning:Connected successfully to mysqueezebox.com. <LF>"
>
> Request: "setsncredentials me@mysqueezebox.com thisisnogood <LF>"
>
> Response: "setsncredentials me@mysqueezebox.com thisisnogood validated:0 warning:Invalid mysqueezebox.com username or password. <LF>"

debug <debug category> <OFF|FATAL|ERROR|WARN|INFO|DEBUG|?|>

The "debug" command allows the caller to query or set the server's internal debugging categories. Use 'OFF' to silence, 'FATAL' for only seeing fatal errors, 'ERROR' for non-fatal errors, etc. Finally, using ? will query the current level for the category. Valid categories can be found under Settings -> Debugging.

Examples:

> Request: "debug d_files ?<LF>"
>
> Response: "debug d_files 0<LF>"
>
> Request: "debug d_itunes 0<LF>"
>
> Response: "debug d_itunes 0<LF>"
>
> Request: "debug d_stream 1<LF>"
>
> Response: "debug d_stream 1<LF>"
>
> Request: "debug d_stream<LF>"
>
> Response: "debug d_stream 0<LF>"

exit

The "exit" command closes the TCP connection with the server and terminates the Command Line Interface session.

Example:

> Request: "exit<LF>"
>
> Response: "exit<LF>"
>
> (Connection terminated)

shutdown

The "shutdown" command shuts down the server. The CLI connection is terminated. Note that, obviously, there is no symmetrical command to restart the server.

Example:

> Request: "shutdown<LF>"
>
> Response: "shutdown<LF>"
>
> (Connection terminated)

Player commands and queries

- player count ?
- player id ?
- player uuid ?
- player name ?

- player ip ?
- player model ?
- player isplayer ?
- player displaytype ?
- player canpoweroff ?
- signalstrength ?
- name
- connected ?
- sleep
- sync
- syncgroups
- power
- mixer volume
- mixer muting
- mixer bass
- mixer treble
- mixer pitch
- show
- display
- linesperscreen
- display ? ?
- displaynow ? ?
- playerpref
- button
- ir
- irenable
- connect
- client forget
- disconnect
- players

player count ?

The "player count ?" query returns the number of players connected to the server.

Example:

Request: "player count ?<LF>"

Response: "player count 2<LF>"

player id <playerindex> ?

The "player id ?" query returns the unique identifier of a player, (<playerid> parameter of many CLI commands). For physical players this is generally the MAC address. The IP address is used for remote streams.

Example:

Request: "player id 0 ?<LF>" (or) "0 player id ?"

Response: "player id 0 04:20:00:12:23:45<LF>"

player uuid <playerindex> ?

The "player uuid ?" query returns the player uuid. The uuid is used by mysqueezebox.com.

Example:

Request: "player uuid 0 ?<LF>" (or) "0 player uuid ?"

Response: "player uuid 0 01234567890123456789012345678901 2<LF>"

player name <playerindex|playerid> ?

The "player name ?" query returns the human-readable name for the specified player. If the name has not been specified by the user

in the Player Settings, then a default name will be used, usually the IP address.

Example:

Request: "player name 0 ?<LF>" or "0 player name ?"

Response: "player name 0 Living Room<LF>"

player ip <playerindex|playerid> ?

The "player ip ?" query returns the IP address (along with port number) of the specified player.

Example:

Request: "player ip 0 ?<LF>" or "0 player ip ?"

Response: "player ip 0 192.168.1.22:3483<LF>"

player model <playerindex|playerid> ?

The "player model ?" query returns the model of the player, currently one of "transporter", "squeezebox2", "squeezebox", "slimp3", "softsqueeze", or "http" (for remote streaming connections).

Example:

Request: "player model 0 ?<LF>" or "0 player model ?"

Response: "player model squeezebox<LF>"

player isplayer <playerindex|playerid> ?

Whether a player is a known player model. Currently know models are "transporter", "squeezebox2", "squeezebox", "slimp3", "softsqueeze", or "http" (for remote streaming connections). Will return 0 for streaming connections.

Example:

Request: "player isplayer 0 ?<LF>" or "0 player isplayer ?"

Response: "player isplayer 1<LF>"

player displaytype <playerindex|playerid> ?

The "player displaytype ?" query returns the display model of the player. Graphical display types start with "graphic-", non-graphical display type with "noritake-".

Example:

Request: "player displaytype 0 ?<LF>" or "0 player displaytype ?"

Response: "player displaytype 0 noritake-katakana<LF>"

player canpoweroff <playerindex|playerid> ?

Returns wether a player can be powered off or not. Current hardware players and SoftSqueeze would return 1, web clients 0.

Examples:

Request: "player canpoweroff 04:20:00:12:23:45 ?<LF>"

Response: "player canpoweroff 04:20:00:12:23:45 1<LF>"

Request: "player canpoweroff 192.168.0.39 ?<LF>"

Response: "player canpoweroff 192.168.0.39 0<LF>"

<playerid> signalstrength ?

Returns the wireless signal strength for the player, range is 1 to 100. Returns 0 if not connected wirelessly.

Example:

Request: "04:20:00:12:23:45 signalstrength ?<LF>"

Response: "04:20:00:12:23:45 signalstrength 76<LF>"

<playerid> name <newname|?>

Sets the name of the player. You may query the player name by passing in "?" (equivalent to "player name ?".)

Example:

Request: "04:20:00:12:23:45 name ?<LF>"

Response: "04:20:00:12:23:45 name Lightyears<LF>"

Request: "04:20:00:12:23:45 name Buzz<LF>"

Response: "04:20:00:12:23:45 name Buzz<LF>"

<playerid> connected ?

Returns the connected state of the player, 1 or 0 depending on the state of the TCP connection to the player. SLIMP3 players, since they use UDP, always return 1.

Examples:

Request: "04:20:00:12:23:45 connected ?<LF>"

Response: "04:20:00:12:23:45 connected 1<LF>"

<playerid> sleep <number|?>

The "sleep" command specifies a number of seconds to continue playing before powering off the player. You may query the amount of time until the player sleeps by passing in "?".

Examples:

Request: "04:20:00:12:23:45 sleep ?<LF>"

Response: "04:20:00:12:23:45 sleep 105.3<LF>"

Request: "04:20:00:12:23:45 sleep 300<LF>"

Response: "04:20:00:12:23:45 sleep 300<LF>"

<playerid> sync <playerindex|playerid|-|?>

The "sync" command specifies the player to synchronise with the given playerid. The command accepts only one playerindex or playerid. To unsync the player, use the "-" parameter.

Note that in both cases the first <playerid> is the player which is already a member of a sync group. When adding a player to a sync group, the second specified player will be added to the group which includes the first player, if necessary first removing the second player from its existing sync-group.

You may query which players are already synced with this player by passing in a "?" parameter. Multiple playerids are separated by a comma. If the player is not synced, "-" is returned.

Examples:

Request: "04:20:00:12:23:45 sync 1<LF>"

Response: "04:20:00:12:23:45 sync 1<LF>"

Request: "04:20:00:12:23:45 sync ?<LF>"

Response: "04:20:00:12:23:45 sync 04:20:00:12:23:21<LF>"

Request: "04:20:00:12:23:45 sync -<LF>"

Response: "04:20:00:12:23:45 sync -<LF>"

syncgroups ?

The "syncgroups" query returns a comma separated list of sync groups members (IDs and names).

Examples:

Request: "syncgroups ?<LF>"

Response: "syncgroups sync_members:04:20:00:12:23:45,04:20:00:12:34:56
sync_member_names:Living%20Room,Kitchen<LF>"

<playerid> power <0|1|?|>

The "power" command turns the player on or off. Use 0 to turn off, 1 to turn on, ? to query and no parameter to toggle the power state of the player.

For remote streaming connections, the command does nothing and the query always returns 1.

Examples:

Request: "04:20:00:12:23:45 power 1<LF>"

Response: "04:20:00:12:23:45 power 1<LF>"

Request: "04:20:00:12:23:45 power ?<LF>"

Response: "04:20:00:12:23:45 power 1<LF>"

<playerid> mixer volume <0 .. 100|-100 .. +100|?>

The "mixer volume" command returns or sets the current volume setting for the player. The scale is 0 to 100, in real numbers (i.e. 34.5 is valid). If the player is muted, the volume is returned as a negative value. Note that players display a 0 to 40 scale, that is, the 0..100 volume divided by 2,5. Likewise, using the "button" command with "volume_up" or "volume_down" parameters increases or decreases the volume by 2,5.

Examples:

Request: "04:20:00:12:23:45 mixer volume ?<LF>"

Response: "04:20:00:12:23:45 mixer volume 98<LF>"

Request: "04:20:00:12:23:45 mixer volume 25<LF>"

Response: "04:20:00:12:23:45 mixer volume 25<LF>"

Request: "04:20:00:12:23:45 mixer volume +10<LF>"

Response: "04:20:00:12:23:45 mixer volume +10<LF>"

<playerid> mixer muting <0|1|toggle|?|>

The "mixer muting" command mutes or unmutes the player. Use 0 to unmute, 1 to mute, ? to query and no parameter (or 'toggle') to toggle the muting state of the player. Note also the "mixer volume" command returns a negative value if the player is muted.

Example:

Request: "04:20:00:12:23:45 mixer muting<LF>"

Response: "04:20:00:12:23:45 mixer muting<LF>"

<playerid> mixer bass <0 .. 100|-100 .. +100|?>

The "mixer bass" command returns or sets the current bass setting for the player. This is only supported by SliMP3 and SqueezeBox (SB1) players. For more information on the 0 to 100 scale, please refer to the "mixer volume" command.

Example:

Request: "04:20:00:12:23:45 mixer bass ?<LF>"

Response: "04:20:00:12:23:45 mixer bass 98<LF>"

Request: "04:20:00:12:23:45 mixer bass 25<LF>"

Response: "04:20:00:12:23:45 mixer bass 25<LF>"

Request: "04:20:00:12:23:45 mixer bass +10<LF>"

Response: "04:20:00:12:23:45 mixer bass +10<LF>"

<playerid> mixer treble <0 .. 100|-100 .. +100|?>

The "mixer treble" command returns or sets the current treble setting for the player. This is only supported by SliMP3 and SqueezeBox (SB1) players. For more information on the 0 to 100 scale, please refer to the "mixer volume" command.

Example:

Request: "04:20:00:12:23:45 mixer treble ?<LF>"

Response: "04:20:00:12:23:45 mixer treble 98<LF>"

Request: "04:20:00:12:23:45 mixer treble 25<LF>"

Response: "04:20:00:12:23:45 mixer treble 25<LF>"

Request: "04:20:00:12:23:45 mixer treble +10<LF>"

Response: "04:20:00:12:23:45 mixer treble +10<LF>"

<playerid> mixer pitch <80 .. 120|-40 .. +40|?>

The "mixer pitch" command returns or sets the current pitch setting for the player (only supported by SqueezeBox (SB1) players).

Example:

Request: "04:20:00:12:23:45 mixer pitch ?<LF>"

Response: "04:20:00:12:23:45 mixer pitch 98<LF>"

Request: "04:20:00:12:23:45 mixer pitch 80<LF>"

Response: "04:20:00:12:23:45 mixer pitch 80<LF>"

Request: "04:20:00:12:23:45 mixer pitch +10<LF>"

Response: "04:20:00:12:23:45 mixer pitch +10<LF>"

<playerid> show <taggedParameters>

The "show" command displays a message on the player display for a given duration. Various options are provided to customize the appearance of the message (font size, centering). If the mesage is too long to fit on the display, it scrolls.

This command is designed to display the message, and by default temporarily cancels any screensaver and increases the brightness to the maximum value.

This command is only echoed once the message display is done.

Please note the CLI expects parameters to be encoded using percent-style escaping (see above): space is represented by "%20". See the examples.

Accepted tagged parameters:

Tag	Description
line1	First line of the display.
line2	Second line of the display. This is the line used for single line display mode (font = huge).
duration	Time in seconds to display the message; this time does not take into account any scrolling time necessary, which will be performed to its completion. The default is 3 seconds.
brightness	Brightness to use to display the message, either 'powerOn', 'powerOff', 'idle' or a value from 0 to 4. The default value is 4. The display brightness is reset to its configured value after the message.
font	Use value "huge" to have line2 displayed on a large font using the entire display. The actual font used depends on the player model. Otherwise the command uses the standard, 2 lines display font.
centered	Use value "1" to center the lines on the display. There is no scrolling in centered mode.
screen	Screen to display text on. Use to display on transporter second screen, i.e. screen:2

Examples:

Request: "04:20:00:12:23:45 show line1:Hello%20World line2:Second%20line duration:1 centered:1<LF>"

Response: "04:20:00:12:23:45 show line1:Hello%20World line2:Second%20line duration:1 centered:1<LF>"

<playerid> display <line1> <line2> <duration>

The "display" command specifies some text to be displayed on the player screen for a specified amount of time (in seconds). Please note the CLI expects parameters to be encoded using percent-style escaping (see above): space is represented by "%20". See the examples.

Examples:

Request: "04:20:00:12:23:45 display Hello World 5<LF>"

Response: "04:20:00:12:23:45 display Hello World 5<LF>"

Request: "04:20:00:12:23:45 display Hello%20World Second%20Line 5<LF>"

Response: "04:20:00:12:23:45 display Hello%20World Second%20Line 5<LF>"

<playerid> linesperscreen ?

The "linesperscreen" command returns how many lines of text can fit in the display, depending on its current setting or font.

Examples:

Request: "04:20:00:12:23:45 linesperscreen ?<LF>"

Response: "04:20:00:12:23:45 linesperscreen 1<LF>"

<playerid> display ? ?

The "display ? ?" command may be used to obtain the text that is currently displayed on the screen.

Examples:

Request: "04:20:00:12:23:45 display ? ?<LF>"

Response: "04:20:00:12:23:45 display Hello World<LF>"

<playerid> displaynow ? ?

The "displaynow" command provides access to the data currently on the display. This differs from the "display ? ?" command in that it

returns the latest data sent to the display, including any animation, double-size fonts, etc...

Examples:

Request: "04:20:00:12:23:45 displaynow ? ?<LF>"

Response: "04:20:00:12:23:45 display Hello World<LF>"

<playerid> playerpref <prefname|namespace:prefname> <prefvalue|?>

The "playerpref" command allows the caller to set and query the server's internal player-specific preferences values.

If you want to query/set a preference from an other namespace than "server" (eg. a plugin), you'll have to prepend the desired namespace to the prefname.

Examples:

Request: "04:20:00:12:23:45 playerpref doublesize ?"

Response: "04:20:00:12:23:45 playerpref doublesize 1"

Request: "04:20:00:12:23:45 playerpref doublesize 0"

Response: "04:20:00:12:23:45 playerpref doublesize 0"

<playerid> playerpref validate <prefname|namespace:prefname> <prefvalue>

The "playerpref validate" command allows the caller to validate a server's internal player-specific preference value without setting it.

If you want to validate a preference from an other namespace than "server" (eg. a plugin), you'll have to prepend the desired namespace to the prefname.

Examples:

Request: "04:20:00:12:23:45 playerpref validate scrollPause 3"

Response: "04:20:00:12:23:45 playerpref validate scrollPause valid:1"

Request: "04:20:00:12:23:45 playerpref validate scrollRate fast"

Response: "04:20:00:12:23:45 playerpref validate scrollRate valid:0"

<playerid> button <buttoncode>

The "button" command simulates a button press. Valid button codes correspond to the functions defined in the Default.map file.

Example:

Request: "04:20:00:12:23:45 button stop<LF>"

Response: "04:20:00:12:23:45 button stop<LF>"

<playerid> ir <ircode> <time>

The "ir" command simulates an IR code. Valid IR codes are defined in the Default.map file.

Example:

Request: "bd:a5:a9:9b:9d:df ir 768910ef 11073.575<LF>"

Response: "bd:a5:a9:9b:9d:df ir 768910ef 11073.575<LF>"

<playerid> irenable <0|1|?|>

The "irenable" command enables or disables IR processing for the player on or off. Use 0 to disable, 1 to enable, ? to query and no parameter to toggle IR processing of the player.

For remote streaming connections, the command does nothing and the query always returns 1.

Examples:

Request: "04:20:00:12:23:45 irenable 1<LF>"

Response: "04:20:00:12:23:45 irenable 1<LF>"

Request: "04:20:00:12:23:45 irenable ?<LF>"

Response: "04:20:00:12:23:45 irenable 1<LF>"

<playerid> connect <ip|www.mysqueezebox.com|www.test.mysqueezebox.com>

The "connect" command tells a Squeezebox 2 or newer player to connect to a different server address or to mysqueezebox.com.

Supported values are:

- ip - A dotted IP address to connect to.
- www.squeezenetwork.com - Connect to SqueezeNetwork.
- www.test.squeezenetwork.com - Connect to SqueezeNetwork test.

If the player is currently a member of a sync-group, then all players in the sync-group will be instructed to switch to the new server and reestablish the sync-group.

Example:

Request: "bd:a5:a9:9b:9d:df connect 192.168.1.10<LF>"

Response: "bd:a5:a9:9b:9d:df connect 192.168.1.10<LF>"

<playerid> client forget

The "client forget" command deletes the client/player from the server database.

Example:

Request: "bd:a5:a9:9b:9d:df client forget<LF>"

Response: "bd:a5:a9:9b:9d:df client forget<LF>"

disconnect <playerid> <ip|www.mysqueezebox.com|www.test.mysqueezebox.com>

The "disconnect" command tells a Squeezebox 2 or newer player on another server instance to disconnect from its server and connect to us. This is the opposite of "connect", where we tell a player connected to us to connect to a different server.

Supported values are:

- ip - A dotted IP address to connect to.
- www.mysqueezebox.com - Connect to mysqueezebox.com.
- www.test.mysqueezebox.com - Connect to mysqueezebox.com test.

Example:

Request: "disconnect bd:a5:a9:9b:9d:df 192.168.1.10<LF>"

Response: "disconnect bd:a5:a9:9b:9d:df 192.168.1.10<LF>"

players <start> <itemsPerResponse>

The "players" query returns information about all "players" (physical players as well as streaming clients) known by the server.

Accepted tagged parameters:

Tag	Description
playerprefs	Comma separated list of preference values to return (for each player).

Returned tagged parameters:

Tag	Description
count	Number of players known by the server. Equivalent to "player count ?".
For each player:	
playerindex	Player index. Item delimiter.
playerid	Player unique identifier. Equivalent to "player id ?".
ip	Player IP and port. Equivalent to "player ip ?".
name	Player name. Equivalent to "player name ?".
model	Player model. Equivalent to "player model ?".
isplayer	Whether a player is a known player model. Will return 0 for streaming connections. "player isplayer ?".
displaytype	Player display type. Not returned for streaming connections. Equivalent to "player displaytype <playerindex> ?".
canpoweroff	Whether the player can be powered off. This value is false for streaming connections.
connected	Connected state. Equivalent to "<playerid> connected ?".
For each defined pref requested:	
prefName	Preference value. Only if the value is defined. Equivalent to "playerpref prefName ?".

Example:

Request: "players 0 2 prefs:doublesize,idleBrightness<LF>"

Response: "players 0 2 count:2 playerindex:0 playerid:a5:41:d2:cd:cd:05 ip:127.0.0.1:60488 name:127.0.0.1 model:softsqueeze displaytype:graphic-280x16 connected:1 doublesize:0 idleBrightness:2 playerindex:1 playerid:00:04:20:02:00:c8 ip:192.168.1.22:3483 name:Movy model:slimp3 displaytype:noritake-katakana connected:1 doublesize:1 idleBrightness:1<LF>"

Database commands and queries

- rescan
- rescanprogress
- abortscan
- wipecache
- info total genres ?
- info total artists ?
- info total albums ?
- info total songs ?
- genres
- artists
- albums
- years
- mediafolder
- musicfolder
- playlists
- playlists tracks
- playlists new
- playlists rename
- playlists delete
- playlists edit
- songinfo
- titles
- search
- pragma

rescan <|playlists|external|full singlefolder|?>

The "rescan" command causes the server to rescan the entire music library, reloading the music file information. If "playlists" is indicated ("rescan playlists"), only the playlist directory is rescanned. If "external" is requested, the rescan will be performed using the external scanner process instead of the in-process scanner. If "full file://some/path" is defined, then only this path will be scanned.

Issued with a "?", "rescan ?" returns if the server is currently scanning.

Scanning occurs when the server starts and following "rescan" and "wipecache" commands.

Examples:

Request: "rescan<LF>"

Response: "rescan<LF>"

Request: "rescan ?<LF>"

Response: "rescan 1<LF>"

rescanprogress <taggedParameters>

The "rescanprogress" query returns details on the scanning progress.

Accepted tagged parameters:

Tag	Descriptio n

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database, otherwise returned with value 0 and the fields below are not returned.
totaltime	Total elapsed time since the start of the scan, format "hh:mm:ss".
importer	A completion percentage for each importer. Importers include "directory" (Music folder), "playlist"(Playlist folder), "iTunes" (iTunes), "musicip" (MusicIP), as well as more technical ones such as "mergeva" (Various Artists merging) and "dboptimize" (Database optimization). The type, quantity and order of importers is determined dynamically as rescan progresses.
info	Additional information about the current scanning step, like eg. the currently scanned file name.
steps	Scanning steps in the order they've been executed.
lastscanfailed	Information about a possible failure in case a scan has not finished in an attended manner.

Example:

Request: "rescanprogress<LF>"

Response: "rescanprogress rescan:1 totaltime:00:00:07 directory:100 playlist:100 itunes:15 <LF>"

Request: "rescanprogress<LF>"

Response: "rescanprogress rescan:1 totaltime:00:01:04 directory:100 playlist:100 itunes:100 itunes_playlists:100 mergeva:100 cleanup1:100 cleanup2:100 dboptimize:47 <LF>"

abortscan

The "abortscan" command causes the server to cancel a running scan. Please note that after stopping a scan this way you'll have to fully rescan your music collection to get consistent data.

Examples:

Request: "abortscan<LF>"

Response: "abortscan<LF>"

wipecache

The "wipecache" command allows the caller to have the server rescan its music library, reloading the music file information. This differs from the "rescan" command in that it first clears the tag database. During a rescan triggered by "wipecache", "rescan ?" returns

true.

Examples:

Request: "wipecache<LF>"

Response: "wipecache<LF>"

info total genres ?

The "info total genres ?" query returns the number of unique genres in the server music database.

Examples:

Request: "info total genres ?<LF>"

Response: "info total genres 18<LF>"

info total artists ?

The "info total artists ?" query returns the number of unique artists in the server music database. The "Composer, band and orchestra in artists" preference (Server Settings, Behavior) determines which contributors are considered artists.

Examples:

Request: "info total artists ?<LF>"

Response: "info total artists 18<LF>"

info total albums ?

The "info total albums ?" query returns the number of unique albums in the server music database.

Examples:

Request: "info total albums ?<LF>"

Response: "info total albums 18<LF>"

info total songs ?

The "info total songs ?" query returns the number of unique songs in the server music database.

Examples:

Request: "info total songs ?<LF>"

Response: "info total songs 18<LF>"

genres <start> <itemsPerResponse> <taggedParameters>

The "genres" query returns all genres known by the server.

Note that the server supports multiple genres per track, depending on the "Multiple items in tags" preference (Server Settings, Behavior).

Accepted tagged parameters:

Tag	Description
search	Search string. The search is case insensitive and obeys the "Search Within Words" server parameter.
artist_id	Limit results to those genres proposed by the artist identified by "artist_id".
album_id	Limit results to those genres available on the album identified by "album_id".
track_id	Limit results to the genres of the track identified by "track_id". If present, other filters are ignored.
genre_id	Limit results to the genre identified by "genre_id". If present, other filters are ignored.

year	Limit results to the genres of the tracks of the given "year".
tags	Determines which tags are returned. Each returned tag is identified by a letter (see below). The default value is empty.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
count	Number of results returned by the query. If no filter parameter is present, this is the same value as returned by "info total genres ?".
Z indexList	An array of arrays indicating how many items start with each key letter.
For each genre:	
id	Genre ID. Item delimiter.
genre	Genre name.
s textkey	The genre's "textkey" is the first letter of the sorting key.

Example:

Request: "genres 0 5<LF>"

Response: "genres 0 5 rescan:1 count:6 id:3 genre:Acid%20Jazz id:4 genre:Alternative%20&%20Punk id:5 genre:French id:6 genre:No%20Genre id:7 genre:Pop <LF>"

Request: "genres 0 5 search:unk<LF>"

Response: "genres 0 5 search:unk count:1 id:4 genre:Alternative%20&%20Punk<LF>"

artists <start> <itemsPerResponse> <taggedParameters>

The "artists" query returns all artists known by the server. The results of this query depend in part on the "Compilations" preference (Server Settings, Behavior). The "Various Artists" pseudo-artist appears if the server groups compilations.

Accepted tagged parameters:

Tag	Description
search	Search substring. The search is case insensitive and obeys the "Search Within Words" server parameter.
genre_id	Genre ID, to restrict the results to those artists with songs of that genre.
album_id	Album ID, to restrict the results to those artists with songs of that album.
track_id	Track ID, to restrict the results to the artist of "track_id". If specified, all other filters are ignored.
artist_id	Artist ID, to restrict the results to a single artist. If specified, all other filters are ignored.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see below). The default value is empty.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
count	Number of results returned by the query. If no search string is present, this is the same value as returned by "info total artists ?"
Z indexList	An array of arrays indicating how many items start with each key letter.
For each artist:	
id	Artist ID. Item delimiter.
artist	Artist name.
s textkey	The artist's "textkey" is the first letter of the sorting key.

Example:

Request: "artists 0 5<LF>"

Response: "artists 0 5 count:7 id:2 artist:Anastacia id:3 artist:Calogero id:4 artist:Evanescence id:5 artist:Leftfield%20%26%20Lydon id:18 artist:Llorca <LF>"

Request: "artists 0 5 genre_id:7<LF>"

Response: "artists 0 5 genre_id:7 count:2 id:2 artist:Anastacia id:19 artist:Sarah%20Connor <LF>"

albums <start> <itemsPerResponse> <taggedParameters>

The "albums" query returns all albums known by the server. The results of this query depend in part on the "Group discs" preference (Server Settings, Behavior).

Accepted tagged parameters:

Tag	Description
search	Search substring. The search is case insensitive and obeys the "Search Within Words" server parameter.
genre_id	Genre ID, to restrict the results to those albums with songs of that genre.
artist_id	Artist ID, to restrict the results to those albums by "artist_id".
track_id	Track ID, to restrict the results to the album of "track_id". If specified, all other filters are ignored.
album_id	Album ID, to restrict the results to a single album. If specified, all other filters are ignored.
year	Year, to restrict the results to those albums of that year.
compilation	Compilation, to restrict the results to those albums that are (1) or aren't (0) compilations.
sort	Sort order of the returned list of albums. One of "album", (the default), "new" which replicates the "New Music" browse mode of the web interface, or "artflow" which sorts by artist, year, album for use with artwork-centric interfaces.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see below). The default value is "l".

Returned tagged parameters:

	Tag	Description
	rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
	count	Number of results returned by the query. If no filter is present, this is the same value as returned by "info total albums ?"
Z	indexList	An array of arrays indicating how many items start with each key letter (or year for a year-dominated sort order).
	For each album:	
	id	Album ID. Item delimiter.
l	album	Album name, including the server's added "(N of M)" if the server is set to group multi disc albums together. See tag "title" for the unmodified value.
y	year	Album year. This is determined by the server based on the album tracks.
j	artwork_track_id	Identifier of one of the album tracks, used by the server to display the album's artwork.
t	title	"Raw" album title as found in the album tracks ID3 tags, as opposed to "album". Note that "title" and "album" are identical if the server is set to group discs together.
i	disc	Disc number of this album. Only if the server is not set to group multi-disc albums together.
q	disccount	Number of discs for this album. Only if known.
w	compilation	1 if this album is a compilation.
a	artist	The album artist (depends on server configuration).
S	artist_id	The album artist id (depends on server configuration).
s	textkey	The album's "textkey" is the first letter of the sorting key.
X	album_replay_gain	The album's replay-gain.

Examples:

Request: "albums 0 4<LF>"

Response: "albums 0 4 count:14 id:1 album:Amadeus%20(Disc%201%20of%202) id:4 album:Anastacia id:5

album:Bounce%20[Single] id:6 album:Fallen<LF>"

Request: "albums 0 5 genre_id:7<LF>"

Response: "albums 0 5 genre_id:7 count:2 id:4 album:Anastacia id:5 album:Bounce%20[Single]<LF>"

Request: "albums 0 5 artist_id:19<LF>"

Response: "albums 0 5 artist_id:19 count:1 id:5 album:Bounce%20[Single]<LF>"

years <start> <itemsPerResponse> <taggedParameters>

The "years" query returns all years known by the server.

Accepted tagged parameters:

Tag	Description
year	Return only the specified year.
hasAlbums:1	Return only years which are valid for albums.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
count	Number of results returned by the query. If no filter parameter is present, this is the same value as returned by "info total genres ?".
For each year:	
year	Year. Item delimiter.

Example:

Request: "years 0 5<LF>"

Response: "years 0 5 rescan:1 count:6 year:1985 year:1987 year:1988 year:2002 year:2003 year:2004 <LF>"

musicfolder <start> <itemsPerResponse> <taggedParameters>

mediafolder <start> <itemsPerResponse> <taggedParameters>

The "musicfolder" query returns the content of a given music folder, starting from the top level directory configured in the server.

"mediafolder" is an alias to "musicfolder".

Accepted tagged parameters:

Tag	Description
folder_id	Browses the folder identified by "folder_id".
return_top	If set to 1, and "folder_id" is provided, will return the data about the listed folder instead of the child folders.
url	Browses the folder identified by "url". If the content of "url" did not happen to be in the server database, it is added to it. "url" has precedence over "folder_id" if both are provided.
type	One of "audio", "list", "video", or "image". Select the media type you want to browse in the given folder.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see below). The default value is empty.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
count	Number of results returned by the query.
For each item in the folder:	
id	Track, playlist or folder ID. Item delimiter.

	filename	File name (encoded in UTF-8, not the system locale)
	type	One of "track", "folder", "playlist", or "unknown".
s	textkey	The item's "textkey" is the first letter of the sorting key.
u	url	The item's full URL.

Example:

Request: "musicfolder 0 10<LF>"

Response: "musicfolder 0 10 count:26 id:1 title:03%20Barbie%20Girl.mp3 type:audio id:2 title:12%20-%20If%20I%20Had%20You.mp3 type:audio id:313 title:A-Ha type:dir id:50 title:Test.m3u type:playlist<LF>"

Request: "musicfolder 0 10 folder_id:313<LF>"

Response: "musicfolder 0 10 folder_id:313 count:2 id:335 title:Lifelines type:dir id:336 title:Minor%20Earth%20Major%20Sky type:dir<LF>"

playlists <start> <itemsPerResponse> <taggedParameters>

The "playlists" query returns all playlists known by the server.

Accepted tagged parameters:

Tag	Description
search	Search substring. The search is case insensitive and obeys the "Search Within Words" server parameter.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see below). The default value is empty.

Returned tagged parameters:

	Tag	Description
	rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
	count	Number of results returned by the query.
	For each playlist:	
	id	Playlist ID. Item delimiter.
	playlist	Playlist name
u	url	Playlist file url
s	textkey	The playlist's "textkey" is the first letter of the sorting key.

Example:

Request: "playlists 0 2<LF>"

Response: "playlists 0 2 count:5 id:37 name:Funky%20Beats id:57 name:SUPER<LF>"

Request: "playlists 0 2 search:SUPER tags:u<LF>"

Response: "playlists 0 2 search:SUPER tags:u count:1 id:57 name:SUPER url:playlist:///Volume/path/file.m3u<LF>"

playlists tracks <start> <itemsPerResponse> <taggedParameters>

The "playlists tracks" query returns the tracks of a given playlist.

Accepted tagged parameters:

Tag	Description
playlist_id	Playlist ID as returned by the "playlists" query. This is a mandatory parameter.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see command "songinfo" for a list of possible fields and their identifying letter). The default tags value for this command is "gald".

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.

count	Number of tracks in the playlist.
For each track:	
playlist index	Index (first item is 0) of the track in the playlist. The first returned instance of this field is equal to start. Item seperator.
Tags	Same tags as defined in command "songinfo".

playlists rename <taggedParameters>

This command renames a saved playlist.

Accepted tagged parameters:

Tag	Description
playlist_id	The id of the playlist to rename.
newname	The new name of the playlist (without .m3u).
dry_run	Used to check if the new name is already used by another playlist. The command performs the name check but does not overwrite the existing playlist. If a name conflict occurs, the command will return a "overwritten_playlist_id" parameter.

Returned tagged parameters:

Tag	Description
overwritten_playlist_id	This returns the playlist id of the playlist overwritten (if "dry_run" is 0 or not present) by renaming the playlist. Only present if a playlist is overwritten.

Examples:

Request: "playlists rename playlist_id:22 newname:Hello<LF>"

Response: "playlists rename playlist_id:22 newname:Hello<LF>"

playlists new <taggedParameters>

This command creates an empty saved playlist, to be further manipulated by other commands.

Accepted tagged parameters:

Tag	Description
name	The name of the playlist (without .m3u).

Returned tagged parameters:

Tag	Description
playlist_id	This returns the playlist id of the created playlist.
overwritten_playlist_id	This returns the playlist id of an existing playlist with the same name. The new playlist is not created.

Examples:

Request: "playlists new name:Hello<LF>"

Response: "playlists new name:Hello playlist_id:345<LF>"

playlists delete <taggedParameters>

This command deletes a saved playlist.

Accepted tagged parameters:

Tag	Description
playlist_id	The id of the playlist to delete.

Examples:

Request: "playlists delete playlist_id:22<LF>"

Response: "playlists delete playlist_id:22<LF>"

playlists edit <taggedParameters>

This command manipulates the track content of a saved playlist.

Accepted tagged parameters:

Tag	Description
playlist_id	The id of the playlist to manipulate.
cmd	One of "up", "down", "move", "delete" or "add", in order to move up, down, delete or add a track.
index	For "cmd:up", "cmd:down", "cmd:move" and "cmd:delete" the index of the track to edit.
toindex	For "cmd:move" the new index of the track to be moved.
title	For "cmd:add", the title of the track to add.
url	For "cmd:add", the url of the track to add.

Examples:

Request: "playlists edit cmd:up playlist_id:22 index:3<LF>"

Response: "playlists edit cmd:up playlist_id:22 index:3<LF>"

Request: "playlists edit cmd:add playlist_id:22 title:Song url:file://...<LF>"

Response: "playlists edit cmd:add playlist_id:22 title:Song url:file://...<LF>"

songinfo <start> <itemsPerResponse> <taggedParameters>

The "songinfo" command returns all the information on a song known by the server. Please note that the <start> and <itemsPerResponse> parameters apply to the individual data fields below and not, as they do in other extended CLI queries, to the number or songs (or artists, genres, etc.) returned; the "songinfo" only ever returns information about a single song.

Accepted tagged parameters:

Tag	Description
url	Song path as returned by other CLI commands. This is a mandatory parameter, except if "track_id" is provided.
track_id	Track ID as returned by other CLI commands. This is a mandatory parameter, except if "url" is provided.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see below). The default value is all info except the url (u) and the multi-valued tags for genre(s) (G & P) and artists (A & S)

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
count	Number of results returned by the query, that is, total number of elements to return for this song.
id	Track ID.
title	Song title
a artist	Artist name.
A <role>	For every artist role (one of "artist", "composer", "conductor", "band", "albumartist" or "trackartist"), a comma separated list of names.
B buttons	A hash with button definitions. Only available for certain plugins such as Pandora.
c coverid	coverid to use when constructing an artwork URL, such as /music/$coverid/cover.jpg
C compilation	1 if the album this track belongs to is a compilation
d duration	Song duration in seconds.
e album_id	Album ID. Only if known.
f filesize	Song file length in bytes. Only if known.
g genre	Genre name. Only if known.
G genres	Genre names, separated by commas (only useful if the server is set to handle multiple items in tags).

i	disc	Disc number. Only if known.
I	samplesize	Song sample size (in bits)
j	coverart	1 if coverart is available for this song. Not listed otherwise.
J	artwork_track_id	Identifier of the album track used by the server to display the album's artwork. Not listed if artwork is not available for this album.
k	comment	Song comments, if any.
K	artwork_url	A full URL to remote artwork. Only available for certain plugins such as Pandora and Rhapsody.
l	album	Album name. Only if known.
L	info_link	A custom link to use for trackinfo. Only available for certain plugins such as Pandora.
m	bpm	Beats per minute. Only if known.
M	musicmagic_mixable	1 if track is mixable, otherwise 0.
n	modificationTime	Date and time song file was last changed on disk.
N	remote_title	Title of the internet radio station.
o	type	Content type. Only if known.
p	genre_id	Genre ID. Only if known.
P	genre_ids	Genre IDs, separated by commas (only useful if the server is set to handle multiple items in tags).
D	addedTime	Date and time song file was first added to the database.
U	lastUpdated	Date and time song file was last updated in the database.
q	disccount	Number of discs. Only if known.
r	bitrate	Song bitrate. Only if known.
R	rating	Song rating, if known and greater than 0.
s	artist_id	Artist ID.
S	<role>_ids	For each role as defined above, the list of ids.
t	tracknum	Track number. Only if known.
T	samplerate	Song sample rate (in KHz)
u	url	Song file url.
v	tagversion	Version of tag information in song file. Only if known.
w	lyrics	Lyrics. Only if known.
x	remote	If 1, this is a remote track.
X	album_replay_gain	Replay gain of the album (in dB), if any
y	year	Song year. Only if known.
Y	replay_gain	Replay gain (in dB), if any

Example:

Request: "songinfo 0 100 track_id:2<LF>"

Response: "songinfo 0 100 track_id:2 count:26 id:2 title:If%20I%20Had%20You artist:Diana%20Krall duration:297.117 album_id:2 filesize:5952369 genre:Vocal comment:Pianist%2Fvocalist%20Diana%20Krall%20pays%20tribute%20to%20the%20Nat%20King%20Cole%20Trio.... album:All%20for%20You modificationTime:Thursday%2C%20March%201%2C%202007%2C%209:21:58%20PM type:mp3 genre_id:2 bitrate:160kbps%20VBR artist_id:3 tracknum:12 tagversion:ID3v2.3.0 year:1995 samplerate:44100 url:file:%2F%2F%2FUsers%2Ffred%2FPrograms%2FLogitech Media Server%2FMusic%2F12%2520-%2520If%2520I%2520Had%2520You.mp3 <LF>"

titles|songs|tracks <start> <itemsPerResponse> <taggedParameters>

The "titles" command returns all titles known by the server.

Accepted tagged parameters:

Tag	Description
genre_id	Genre ID, to restrict the results to the titles of that genre.
artist_id	Artist ID, to restrict the results to the titles of that artist.
album_id	Album ID, to restrict the results to the titles of that album.
track_id	Track ID, to restrict the results to a single track.
year	Year, to restrict the results to the titles of that year.

search	Search substring. The search is case insensitive and obeys the "Search Within Words" server parameter.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see command "songinfo" for a list of possible fields and their identifying letter). The default tags value for this command is "gald".
sort	Sorting, one of "title" (the default), "tracknum" (in which case the track field ("t") is added automatically to the response) or "albumtrack" (in which case the track and album-title fields ("lt") are added automatically to the response). Sorting by tracks is possible only if tracks are defined and for a single album.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
count	Number of results returned by the query. If no search string or album/artist/genre filter is present, this is the same value as returned by "info total songs ?".
For each title:	
Tags	Same tags as defined in command "songinfo".

Example:

Request: "titles 0 2<LF>"

Response: "titles 0 2 count:100 id:55 title:Ancestral%20Aid genre:Soundtrack artist:Various%20Artists album:The%20Hunt%20For%20Red%20October duration:136.93387755102 disc:1 track:5 year:1990 url:file://Disk/The%20Hunt%20For%20Red%20October/Ancestral%20Aid.mp3 id:1 title:Any%20How genre:Acid%20Jazz artist:Llorca album:New%20Comer duration:340.297142857143 track:5 year:2001 url:...<LF>"

Request: "titles 0 12 album:Anastacia tags:p<LF>"

Response: "titles 0 12 album:Anastacia tags:p count:12 id:4 title:Heavy%20On%20My%20Heart url:... id:1 title:I%20Do url:... id:2 title:Left%20Outside%20Alone url:...<LF>"

search <start> <itemsPerResponse> <taggedParameters>

The "search" command returns artists, albums and tracks matching a search string.

Please note that "start" and "itemsPerResponse" are calculated per category. If you eg. have genres and tracks with the search term in them, you'll get "itemsPerResponse" number of each of them. The total number of items returned therefore can be a multiple of "itemsPerResponse".

Accepted tagged parameters:

Tag	Description
term	Search string

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
count	Total number of results returned by the query. This is the sum of "artists_count", "albums_count" and "tracks_count".
artists_count	Total number of artists found.
albums_count	Total number of albums found.
genres_count	Total number of genres found.
tracks_count	Total number of tracks found.
For each artist:	
artist_id	Artist ID.
artist	Artist name.
For each album:	
album_id	Album ID.
album	Album title.

For each genre:

genre_id	Genre ID.
genre	Genre title.

For each track:

track_id	Track ID.
track	Track title.

Example:

> Request: "search 0 20 term:al<LF>"
>
> Response: "search 0 20 term:al count:9 artists_count:2 albums_count:1 tracks_count:6 artist_id:2 artist:Alanis%20Morissette artist_id:37 artist:Alphaville album_id:10 album:All%20Time%20Greatest%20Hits%20%5BBarry%20White%5D%20MMM track_id:11 track:All%20I%20Really%20Want track_id:68 track:The%20Sun%20Always%20Shines%20On%20TV track_id:185 track:All%20Around%20The%20World track_id:189 track:Give%20Me%20All%20Your%20Love%20%5BSingle%20Cut%5D track_id:245 track:Better%20All%20The%20Time

pragma <pragma string>

The "pragma" command executes the given pragma string against the SQLite database engine. If using MySQL, this command has no effect. For a list of available pragmas, see sqlite.org. Warning: Do not use this function unless you know what you are doing!

Example:

> pragma locking_mode = NORMAL

Playlist commands and queries

- play
- stop
- pause
- mode ?
- time
- genre ?
- artist ?
- album ?
- title ?
- duration ?
- remote ?
- current_title ?
- path ?
- playlist play
- playlist add
- playlist insert
- playlist deleteitem
- playlist move
- playlist delete
- playlist preview
- playlist resume
- playlist save
- playlist loadalbum
- playlist addalbum
- playlist loadtracks
- playlist addtracks
- playlist insertalbum
- playlist deletealbum
- playlist clear
- playlist zap
- playlist name ?
- playlist url ?
- playlist modified ?
- playlist playlistsinfo
- playlist index

- playlist genre ?
- playlist artist ?
- playlist album ?
- playlist title ?
- playlist path ?
- playlist remote ?
- playlist duration ?
- playlist tracks ?
- playlist shuffle
- playlist repeat
- playlistcontrol

<playerid> play <fadeInSecs>

The "play" command allows to start playing the current playlist. The "fadeInSecs" parameter may be passed to specify fade-in period.

Example:

Request: "04:20:00:12:23:45 play<LF>"

Response: "04:20:00:12:23:45 play<LF>"

<playerid> stop

The "stop" command allows to stop playing the current playlist.

Example:

Request: "04:20:00:12:23:45 stop<LF>"

Response: "04:20:00:12:23:45 stop<LF>"

<playerid> pause <0|1|> <fadeInSecs> <suppressShowBriefly>

Example:

You may use "pause 1" to force the player to pause, "pause 0" to force the player to unpause and "pause" to toggle the pause state. The "fadeInSecs" parameter may be passed to specify fade-in period when unpausing. The "showBriefly" parameter may be passed to specify not to show a pause icon on squeezeplay-based devices (as is the case with hitting 'power off' on the SBController, which pauses play but should not display an icon, see bug 13521)

Request: "04:20:00:12:23:45 pause<LF>"

Response: "04:20:00:12:23:45 pause<LF>"

<playerid> mode ?

The "mode" command allows to query the player state and returns one of "play", "stop" or "pause". If the player is off, "mode ?" returned value is undefined.

Example:

Request: "04:20:00:12:23:45 mode ?<LF>"

Response: "04:20:00:12:23:45 mode stop<LF>"

<playerid> time <number|-number|+number|?>

The "time" command allows you to query the current number of seconds that the current song has been playing by passing in a "?". You may jump to a particular position in a song by specifying a number of seconds to seek to. You may also jump to a relative position within a song by putting an explicit "-" or "+" character before a number of second you would like to seek.

Examples:

Request: "04:20:00:12:23:45 time ?<LF>"

Response: "04:20:00:12:23:45 time 12.55<LF>"

Request: "04:20:00:12:23:45 time 5<LF>"

Response: "04:20:00:12:23:45 time 5<LF>"

Request: "04:20:00:12:23:45 time -5<LF>"

Response: "04:20:00:12:23:45 time -5<LF>"

<playerid> genre ?

<playerid> artist ?

<playerid> album ?

<playerid> title ?

<playerid> duration ?

<playerid> remote ?

<playerid> current_title ?

<playerid> path ?

The "genre", "artist", "album", "title" "duration", "remote", "current_title" and "path" commands allow for querying information about the song currently playing. "remote" returns 1 if the current song is a remote stream. "current_title" returns the current title for remote streams or the song title as formatted for the player.

Examples:

Request: "04:20:00:12:23:45 genre ?<LF>"

Response: "04:20:00:12:23:45 genre Rock<LF>"

Request: "04:20:00:12:23:45 artist ?<LF>"

Response: "04:20:00:12:23:45 artist Abba<LF>"

Request: "04:20:00:12:23:45 album ?<LF>"

Response: "04:20:00:12:23:45 album Greatest%20Hits<LF>"

Request: "04:20:00:12:23:45 title ?<LF>"

Response: "04:20:00:12:23:45 title Voulez%20vous<LF>"

Request: "04:20:00:12:23:45 duration ?<LF>"

Response: "04:20:00:12:23:45 duration 103.2<LF>"

Request: "04:20:00:12:23:45 remote ?<LF>"

Response: "04:20:00:12:23:45 remote 0<LF>"

Request: "04:20:00:12:23:45 current_title ?<LF>"

Response: "04:20:00:12:23:45 current_title 1-Voulez%20vous%20(ABBA)<LF>"

Request: "04:20:00:12:23:45 path ?<LF>"

Response: "04:20:00:12:23:45 path pathtofile<LF>"

<playerid> playlist play <item> <title> <fadeInSecs>

The "playlist play" command puts the specified song URL, playlist or directory contents into the current playlist and plays starting at the first item. Any songs previously in the playlist are discarded. An optional title value may be passed to set a title. This can be useful for remote URLs. The "fadeInSecs" parameter may be passed to specify fade-in period.

Examples:

Request: "04:20:00:12:23:45 playlist play /music/abba/01_Voulez_Vous.mp3<LF>"

Response: "04:20:00:12:23:45 playlist play /music/abba/01_Voulez_Vous.mp3<LF>"

<playerid> playlist add <item>

The "playlist add" command adds the specified song URL, playlist or directory contents to the end of the current playlist. Songs currently playing or already on the playlist are not affected.

Examples:

Request: "04:20:00:12:23:45 playlist add /music/abba/01_Voulez_Vous.mp3<LF>"

Response: "04:20:00:12:23:45 playlist add /music/abba/01_Voulez_Vous.mp3<LF>"

Request: "04:20:00:12:23:45 playlist add /playlists/abba.m3u<LF>"

Response: "04:20:00:12:23:45 playlist add /playlists/abba.m3u<LF>"

<playerid> playlist insert <item>

The "playlist insert" command inserts the specified song URL, playlist or directory contents to be played immediately after the current song in the current playlist. Any songs currently playing or already on the playlist are not affected.

Examples:

Request: "04:20:00:12:23:45 playlist insert /music/abba/01_Voulez_Vous.mp3<LF>"

Response: "04:20:00:12:23:45 playlist insert /music/abba/01_Voulez_Vous.mp3<LF>"

Request: "04:20:00:12:23:45 playlist insert /playlists/abba.m3u<LF>"

Response: "04:20:00:12:23:45 playlist insert /playlists/abba.m3u<LF>"

<playerid> playlist deleteitem <item>

The "playlist deleteitem" command removes the specified song URL, playlist or directory contents from the current playlist.

Examples:

Request: "04:20:00:12:23:45 playlist deleteitem /music/abba/01_Voulez_Vous.mp3<LF>"

Response: "04:20:00:12:23:45 playlist deleteitem /music/abba/01_Voulez_Vous.mp3<LF>"

<playerid> playlist move <fromindex> <toindex>

The "playlist move" command moves the song at the specified index to a new index in the playlist. An offset of zero is the first song in the playlist.

Examples:

Request: "04:20:00:12:23:45 playlist move 0 5<LF>"

Response: "04:20:00:12:23:45 playlist move 0 5<LF>"

<playerid> playlist delete <songindex>

The "playlist delete" command deletes the song at the specified index from the current playlist.

Examples:

Request: "04:20:00:12:23:45 playlist delete 5<LF>"

Response: "04:20:00:12:23:45 playlist delete 5<LF>"

<playerid> playlist preview <taggedParameters>

When called without a cmd param of stop, replace the current playlist with the playlist specified by url, but save the current playlist to tempplaylist_<playerid>.m3u for later retrieval. When called with the cmd param of stop, stops the currently playing playlist and loads (if possible) the previous playlist. Restored playlist jumps to beginning of CURTRACK when present in m3u file, and does not autoplay restored playlist.

Examples:

> Request: "04:20:00:12:23:45 playlist preview url:db:album.titlesearch=A%20FEAST%20OF%20WIRE title:A%20Feast%20Of%20Wire<LF>"
>
> Response: "04:20:00:12:23:45 playlist preview url:db:album.titlesearch=A%20FEAST%20OF%20WIRE title:A%20Feast%20Of%20Wire<LF>"
>
> Request: "04:20:00:12:23:45 playlist preview cmd:stop<LF>"
>
> Response: "04:20:00:12:23:45 playlist preview cmd:stop<LF>"

<playerid> playlist resume <playlist> <taggedParameters>

Replace the current playlist with the playlist specified by p2, starting at the song that was playing when the file was saved. (Resuming works only with M3U files saved with the "playlist save" command below.) Shortcut: use a bare playlist name (without leading directories or trailing .m3u suffix) to load a playlist in the saved playlists folder.

Optional tagged parameters are noplay, which when non-zero will not auto-start the track, and wipePlaylist, which will destroy the saved playlist from both the filesystem and from the DB (these tagged params are typically used for resuming a temporarily cached playlist, e.g. after exiting alarm sound preview on squeezeplay devices).

Examples:

> Request: "04:20:00:12:23:45 playlist resume abba<LF>"
>
> Response: "04:20:00:12:23:45 playlist resume abba<LF>"

<playerid> playlist save <filename> <taggedParameters>

Saves a playlist file in the saved playlists directory. Accepts a playlist filename (without .m3u suffix) and saves in the top level of the playlists directory.

Optional tagged param silent. When non-zero, suppresses any showBriefly displayed.

Examples:

> Request: "04:20:00:12:23:45 playlist save abba<LF>"
>
> Response: "04:20:00:12:23:45 playlist save abba<LF>"

<playerid> playlist loadalbum <genre> <artist> <album>

The "playlist loadalbum" command puts songs matching the specified genre artist and album criteria on the playlist. Songs previously in the playlist are discarded.

Examples:

> Request: "04:20:00:12:23:45 playlist loadalbum Rock Abba *<LF>"
>
> Response: "04:20:00:12:23:45 playlist loadalbum Rock Abba *<LF>"

<playerid> playlist addalbum <genre> <artist> <album>

The "playlist addalbum" command appends all songs matching the specified criteria onto the end of the playlist. Songs currently playing or already on the playlist are not affected.

Examples:

Request: "04:20:00:12:23:45 playlist addalbum Rock Abba *<LF>"

Response: "04:20:00:12:23:45 playlist addalbum Rock Abba *<LF>"

<playerid> playlist loadtracks <searchparam>

The "playlist loadtracks" command puts tracks matching the specified query on the playlist. Songs previously in the playlist are discarded. Note: you must provide a particular form to the searchparam (see examples)

Examples:

Request: "04:20:00:12:23:45 playlist loadtracks track.titlesearch=purple <LF>"

Response: "04:20:00:12:23:45 playlist loadtracks track.titleseach=purple <LF>"

Request: "04:20:00:12:23:45 playlist loadtracks album.titlesearch=3121 <LF>"

Response: "04:20:00:12:23:45 playlist loadtracks album.titlesearch=3121 <LF>"

Request: "04:20:00:12:23:45 playlist loadtracks contributor.namesearch=prince <LF>"

Response: "04:20:00:12:23:45 playlist loadtracks contributor.namesearch=prince <LF>"

<playerid> playlist addtracks <searchparam>

The "playlist addtracks" command appends all songs matching the specified criteria onto the end of the playlist. Songs currently playing or already on the playlist are not affected. Note: you must provide a particular form to the searchparam (see examples)

Examples:

Request: "04:20:00:12:23:45 playlist addtracks track.titlesearch=purple <LF>"

Response: "04:20:00:12:23:45 playlist addtracks track.titleseach=purple <LF>"

Request: "04:20:00:12:23:45 playlist addtracks album.titlesearch=3121 <LF>"

Response: "04:20:00:12:23:45 playlist addtracks album.titlesearch=3121 <LF>"

Request: "04:20:00:12:23:45 playlist addtracks contributor.namesearch=prince <LF>"

Response: "04:20:00:12:23:45 playlist addtracks contributor.namesearch=prince <LF>"

<playerid> playlist insertalbum <genre> <artist> <album>

The "playlist insertalbum" command inserts all songs matching the specified criteria at the top of the playlist. Songs already on the playlist are not affected.

Examples:

Request: "04:20:00:12:23:45 playlist addalbum Rock Abba *<LF>"

Response: "04:20:00:12:23:45 playlist addalbum Rock Abba *<LF>"

<playerid> playlist deletealbum <genre> <artist> <album>

The "playlist deletealbum" command removes songs matching the specified genre artist and album criteria from the playlist.

Examples:

Request: "04:20:00:12:23:45 playlist deletealbum Rock Abba *<LF>"

Response: "04:20:00:12:23:45 playlist deletealbum Rock Abba *<LF>"

<playerid> playlist clear

The "playlist clear" command removes any song that is on the playlist. The player is stopped.

Examples:

Request: "04:20:00:12:23:45 playlist clear<LF>"

Response: "04:20:00:12:23:45 playlist clear<LF>"

<playerid> playlist zap <songindex>

The "playlist zap" command adds the song at index songindex into the zapped song playlist.

Examples:

Request: "04:20:00:12:23:45 playlist zap 3<LF>"

Response: "04:20:00:12:23:45 playlist zap 3<LF>"

<playerid> playlist name ?

The "playlist name" command returns the name of the saved playlist last loaded into the Now Playing playlist, if any.

Examples:

Request: "04:20:00:12:23:45 playlist name ?<LF>"

Response: "04:20:00:12:23:45 playlist name Jazz%20Favorites <LF>"

<playerid> playlist url ?

The "playlist url" command returns the URL of the saved playlist last loaded into the Now Playing playlist, if any.

Examples:

Request: "04:20:00:12:23:45 playlist url ?<LF>"

Response: "04:20:00:12:23:45 playlist url file:///Users/dean/Music/testmusic/Zapped%20Songs.m3u<LF>"

<playerid> playlist modified ?

The "playlist modified" returns the modification state of the saved playlist last loaded into the Now Playing playlist, if any. If "1", the playlist has been modified since it was loaded.

Examples:

Request: "04:20:00:12:23:45 playlist modified ?<LF>"

Response: "04:20:00:12:23:45 playlist modified 0<LF>"

<playerid> playlist playlistsinfo <taggedParameters>

The "playlist playlistsinfo" query returns information on the saved playlist last loaded into the Now Playing playlist, if any.

Accepted tagged parameters:

Tag	Descriptio n

Returned tagged parameters:

Tag	Description
id	Playlist id.
name	Playlist name. Equivalent to "playlist name ?".
modified	Modification state of the saved playlist. Equivalent to "playlist modified ?".
url	Playlist url. Equivalent to "playlist url ?".

Example:

Request: "a5:41:d2:cd:cd:05 playlist playlistsinfo <LF>"

Response: "a5:41:d2:cd:cd:05 playlist playlistsinfo id:267 name:A98 modified:0 url:file://Volumes/... <LF>"

<playerid> playlist index <index|+index|-index|?> <fadeInSecs>

The "playlist index" command sets or queries the song that is currently playing by index. When setting, a zero-based value may be used to indicate which song to play. An explicitly positive or negative number may be used to jump to a song relative to the currently playing song. The index can only be set if the playlist is not empty. If an index parameter is set then "fadeInSecs" may be passed to specify fade-in period. The value of the current song index may be obtained by passing in "?" as a parameter.

Examples:

Request: "04:20:00:12:23:45 playlist index +1<LF>"

Response: "04:20:00:12:23:45 playlist index +1<LF>"

Request: "04:20:00:12:23:45 playlist index 5<LF>"

Response: "04:20:00:12:23:45 playlist index 5<LF>"

Request: "04:20:00:12:23:45 playlist index ?<LF>"

Response: "04:20:00:12:23:45 playlist index 5<LF>"

<playerid> playlist genre <index> ?

<playerid> playlist artist <index> ?

<playerid> playlist album <index> ?

<playerid> playlist title <index> ?

<playerid> playlist path <index> ?

<playerid> playlist remote <index> ?

<playerid> playlist duration <index> ?

The "playlist genre", "playlist artist", "playlist album", "playlist title", "playlist path", "playlist remote" and "playlist duration" queries return the requested information for a given song at an index position in the current playlist. "playlist remote" returns 1 if the "song" is a remote stream.

Examples:

Request: "04:20:00:12:23:45 playlist genre 3 ?<LF>"

Response: "04:20:00:12:23:45 playlist genre 3 Rock<LF>"

Request: "04:20:00:12:23:45 playlist artist 3 ?<LF>"

Response: "04:20:00:12:23:45 playlist artist 3 Abba<LF>"

Request: "04:20:00:12:23:45 playlist album 3 ?<LF>"

Response: "04:20:00:12:23:45 playlist album 3 Greatest Hits<LF>"

Request: "04:20:00:12:23:45 playlist title 3 ?<LF>"

Response: "04:20:00:12:23:45 playlist title 3 Voulez Vous<LF>"

Request: "04:20:00:12:23:45 playlist path 3 ?<LF>"

Response: "04:20:00:12:23:45 playlist path 3 file:///Volumes/Music/ABBA/...<LF>"

Request: "04:20:00:12:23:45 playlist remote 3 ?<LF>"

Response: "04:20:00:12:23:45 playlist remote 3 0<LF>"

Request: "04:20:00:12:23:45 playlist duration 3 ?<LF>"

Response: "04:20:00:12:23:45 playlist duration 3 103.2<LF>"

<playerid> playlist tracks ?

The "playlist tracks" command returns the the total number of tracks in the current playlist

Example:

Request: "04:20:00:12:23:45 playlist tracks ?<LF>"

Response: "04:20:00:12:23:45 playlist tracks 7<LF>"

<playerid> playlist shuffle <0|1|2|?|>

The "playlist shuffle" command is used to shuffle, unshuffle or query the shuffle state for the current playlist. A value of "0" indicates that the playlist is not shuffled, "1" indicates that the playlist is shuffled by song, and "2" indicates that the playlist is shuffled by album. Used with no parameter, the command toggles the shuffling state.

Example:

Request: "04:20:00:12:23:45 playlist shuffle ?<LF>"

Response: "04:20:00:12:23:45 playlist shuffle 1<LF>"

Request: "04:20:00:12:23:45 playlist shuffle 0<LF>"

Response: "04:20:00:12:23:45 playlist shuffle 0<LF>"

<playerid> playlist repeat <0|1|2|?|>

The "playlist repeat" command is used to indicate or query if the player will stop playing at the end of the playlist, repeat the current song indefinitely, or repeat the current playlist indefinitely. A value of "0" indicates that the player will stop at the end of the playlist, "1" indicates that the player will repeat the current song indefinitely and a value of "2" indicates that the player will repeat the entire playlist indefinitely. Used with no parameter, the command toggles the repeat state.

Example:

Request: "04:20:00:12:23:45 playlist repeat ?<LF>"

Response: "04:20:00:12:23:45 playlist repeat 2<LF>"

Request: "04:20:00:12:23:45 playlist repeat 0<LF>"

Response: "04:20:00:12:23:45 playlist repeat 0<LF>"

<playerid> playlistcontrol <taggedParameters>

The "playlistcontrol" command enables playlist operations using IDs as returned by extended CLI queries (titles, artists, playlists, etc).

Accepted tagged parameters:

Tag	Description
cmd	Command to perform on the playlist, one of "load", "add", "insert" or "delete". This parameter is mandatory. If no additional parameter is provided, the entire DB is loaded/added/inserted/deleted.
genre_id	Genre ID, to restrict the results to the titles of that genre.
artist_id	Artist ID, to restrict the results to the titles of that artist.
album_id	Album ID, to restrict the results to the titles of that album.
track_id	Comma-separated list of track IDs, to restrict the results to these track_ids. If this parameter is provided, then any genre_id, artist_id and/or album_id parameter is ignored. The tracks are added to the playlist in the given order.
year|year_id	Year, to restrict the results to the given year. The form year_id is accepted for backwared compatibility but is deprecated.
playlist_id	Playlist ID, to restrict the results to this playlist_id. If this parameter is provided, then any genre_id, artist_id, album_id and/or track_id parameter is ignored.
folder_id	Folder ID, to restrict the results to files in this folder_id. If this parameter is provided,

	then any all the others are ignored. Note that "cmd:delete" is not supported for folders.
playlist_name	Playlist name, to restrict the results to this playlist_name. If this parameter is provided, then any genre_id, artist_id, album_id, track_id and/or playlist_id parameter is ignored.
play_index	If this parameter is provided along with "cmd:load" then playback will start with the indicated track.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The command may therefore have missed items. Not returned if no scan is in progress.
count	Number of elements loaded/added/inserted or max number of elements deleted. For folders, only 1 is returned.

Example:

Request: "a5:41:d2:cd:cd:05 playlistcontrol cmd:add genre_id:9<LF>"

Response: "a5:41:d2:cd:cd:05 playlistcontrol cmd:add genre_id:9 count:33<LF>"

Request: "a5:41:d2:cd:cd:05 playlistcontrol cmd:load album_id:22<LF>"

Response: "a5:41:d2:cd:cd:05 playlistcontrol cmd:load album_id:22 count:12<LF>"

Compound queries

- serverstatus
- status
- displaystatus
- readdirectory

serverstatus <start> <itemsPerResponse> <taggedParameters>

The "serverstatus" query returns a complete status about the server, including its players.

Clients can subscribe to "serverstatus" queries, so that the query results are automatically returned asynchronously whenever a change occurs to the server. Please note this mechanism is completely distinct from the "listen" and "subscribe" commands described elsewhere in this document.

Accepted tagged parameters:

Tag	Description
prefs	Comma separated list of server preference values to return.
playerprefs	Comma separated list of player preference values to return (for each player).
subscribe	This optional parameter controls the subscription to the server status. Only one status subscription is possible per connection. Subscription is enabled by using this parameter with a positive integer. It is disabled by using "-". When the subscription is enabled, normal "serverstatus" queries (i.e. not using the "subscribe" parameter) can be performed and will have no effect on the subscription in place. When enabled, the "serverstatus" query is automatically re-generated on server change (and sent asynchronously to the CLI client). The number indicates the time interval in seconds between automatic generations in case nothing happened to the server info in the interval. Use "0" to disable this last feature and only be notified on changes. Please see the example. Some situations will lead to multiple status queries generated very close to another. This is a limitation of the change detection "algorithm". Please note this mechanism is completely distinct from the "listen" and "subscribe" commands described above.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may

	therefore be incomplete. Not returned if no scan is in progress.
lastscan	Returned with the timestamp when the last scan finished. Not returned if no scan has been run yet.
progressname	Returned with the name for for current scan phase. Not returned ifno scan is in progress.
progressdone	Returned with the current value of items completed for current scan phase. Not returned if no scan is in progress.
progresstotal	Returned with the total value of items found for current scan phase. Not returned if no scan is in progress.
lastscanfailed	Information about a possible failure in case a scan has not finished in an attended manner.
version	Logitech Media Server version. Equivalent to "version ?"
info total albums	Number of albums known to the server. Equivalent to "info total albums ?"
info total artists	Number of artists known to the server. Equivalent to "info total artists ?"
info total genres	Number of genres known to the server. Equivalent to "info total genres ?"
info total songs	Number of songs known to the server. Equivalent to "info total songs ?"
For each defined pref requested:	
prefName	Preference value. Only if the value is defined. Equivalent to "pref prefName ?".
player count	Number of players known by the server. Equivalent to "player count ?".
For each player:	Essentially, this list is equivalent to the one returned by "players".
playerid	Player unique identifier. Item delimiter. Equivalent to "player id ?".
uuid	Player unique identifier. Equivalent to "player uuid ?".
ip	Player IP and port. Equivalent to "player ip ?".
name	Player name. Equivalent to "player name ?".
model	Whether the player is powered on or off.
power	Player model. Equivalent to "player model ?".
isplayer	Whether a player is a known player model. Will return 0 for streaming connections. Equivalent to "player isplayer ?".
displaytype	Player display type. Not returned for streaming connections. Equivalent to "player displaytype <playerindex> ?".
canpoweroff	Whether the player can be powered off. This value is false for streaming connections.
connected	Connected state. Equivalent to "<playerid> connected ?".
player_needs_upgrade	Connected player needs a firmware upgrade.
player_is_upgrading	Connected player is in the process of performing a firmware update.
sn player count	Number of players connected to the current mysqueezebox.com account.
For each player connected to mysqueezebox.com:	
id	Player unique identifier on mysqueezebox.com.
playerid	Player unique identifier (MAC address)..
name	Player name.
other player count	Number of players connected to other discovered servers in the local network.
For each player connected to some other server in the local network:	
playerid	Player unique identifier (MAC address)..
name	Player name.
server	The server to which the player is connected
model	Player model. Please note that only Squeezebox2 and later can remotely be disconnected.
For each defined player pref requested:	
prefName	Preference value. Only if the value is defined. Equivalent to "playerpref prefName ?".

Example:

<playerid> status <start> <itemsPerResponse> <taggedParameters>

The "status" query returns the complete status about a given player, including the current playlist. Set the <start> parameter to "-" to

get the playlist data starting from the current song.

In this "curent" mode and if repeat is on, the server will attempt to return <itemsPerResponse> elements, by repeating the playlist at most once, unless shuffling is on and the server is configured to re-shuffle the playlist at each loop (in which case it is impossible to predict the song following the last one in the playlist until this last song has finished playing).

Similarly, in the "curent" mode, if repeat is one, only the current song is returned, regardless of the value of <itemsPerResponse>.

Clients can subscribe to "status" queries, so that the query results are automatically returned asynchronously whenever a change occurs to a player. Please note this mechanism is completely distinct from the "listen" and "subscribe" commands described elsewhere in this document.

Accepted tagged parameters:

Tag	Description
tags	Determines which tags are returned. Each returned tag is identified by a letter (see command "songinfo" for a list of possible fields and their identifying letter). The default tags value for this query is "gald".
subscribe	This optional parameter controls the subscription to the player status. Only one status subscription is possible per player and connection. Subscription is enabled by using this parameter with a positive integer. It is disabled by using "-". When the subscription is enabled, normal "status" queries (i.e. not using the "subscribe" parameter) can be performed and will have no effect on the subscription in place. When enabled, the "status" request is automatically re-generated on player change (and sent asynchronously to the CLI client). The number indicates the time interval in seconds between automatic generations in case nothing happened to the player in the interval. Use "0" to disable this last feature and only be notified on player or playlist changes. Please see the example. Some situations will lead to multiple status queries generated very close to another. This is a limitation of the change detection "algorithm". If the player is manually (through the web page) or automatically deleted from the server, the status query returns the "error" tag with value "invalid player" and the subscription is terminated. Please note this mechanism is completely distinct from the "listen" and "subscribe" commands described above.

Returned tagged parameters:

Tag	Description
rescan	Returned with value 1 if the server is still scanning the database. The results may therefore be incomplete. Not returned if no scan is in progress.
error	Returned with value "invalid player" if the player this subscription query referred to does no longer exist. In non subscription mode, the query simply echoes itself (i.e. produces no result) if <playerid> is wrong.
player_name	Name of the player.
player_connected	Connected state of the player.
player_needs_upgrade	Connected player needs a firmware upgrade.
player_is_upgrading	Connected player is in the process of performing a firmware update.
power	Power state of the player. Not returned for remote streaming connections.
signalstrength	Signal strength (only for Squeezeboxen and Transporters).
If player is on:	
mode	Player mode.
remote	Returns 1 if a remote stream is currently playing.
current_title	Returns the current title for remote streams. Only if remote stream is playing.
time	Elapsed time into the current song. Decimal seconds. Only if current song.
rate	Player rate. Only if there is a current song.
duration	Duration of the current song. Decimal seconds. Only if current song and if the duration is known (it is not for remote streams).
sleep	If set to sleep, the amount (in seconds) it was set to.
will_sleep_in	Seconds left until sleeping. Only if set to sleep.

sync_master	ID of the master player in the sync group this player belongs to. Only if synced.
sync_slaves	Comma-separated list of player IDs, slaves to sync_master in the sync group this player belongs to. Only if synced.
mixer volume	Not returned for remote streaming connections.
mixer treble	Only for SliMP3 and Squeezebox1 players.
mixer bass	Only for SliMP3 and Squeezebox1 players.
mixer pitch	Only for Squeezebox1 players.
playlist repeat	0 no repeat, 1 repeat song, 2 repeat playlist.
playlist shuffle	0 no shuffle, 1 shuffle songs, 2 shuffle albums.
playlist_id	Playlist id, if the current playlist is a stored playlist.
playlist_name	Playlist name, if the current playlist is a stored playlist. Equivalent to "playlist name ?".
playlist_modified	Modification state of the saved playlist (if the current playlist is one). Equivalent to "playlist modified ?".
playlist_timestamp	Timestamp of the current playlist, in seconds. Changes to the playlist (insertion/removal/shuffling) result in an increase of this value. This can be used to detect the entire playlist has to be reacquired.
playlist_tracks	Number of tracks in the current playlist. Only if there is a playlist.

If playlist information exist/requested, for each song in the playlist:

playlist index	Index (first item is 0) of the playlist entry in the player playlist. Unless <start> is "-", the first returned instance of this field is equal to start. If <start> is "-", the first returned instance of this field contains the index of the currently playing song in the player playlist. Item seperator.
Tags	Same tags as defined in command "songinfo".

Example:

Simple example

Request: "a5:41:d2:cd:cd:05 status 0 2 tags:<LF>"

Response: "a5:41:d2:cd:cd:05 status 0 2 tags: player_name:127.0.0.1 player_connected:1 power:1 mode:play rate:1 time:13.7129358076728 duration:252.630204081633 mixer%20volume:50 mixer%20treble:50 mixer%20bass:50 mixer%20pitch:100 playlist%20repeat:2 playlist%20shuffle:0 playlist_cur_index:1 playlist_tracks:3 playlist%20index:0 title:Left%20Outside%20Alone playlist%20index:1 title:Bounce%20[Original%20Version]<LF>"

Current mode example

Request: "a5:41:d2:cd:cd:05 status - 2 tags:<LF>"

Response: "a5:41:d2:cd:cd:05 status - 2 tags: player_name:127.0.0.1 player_connected:1 power:1 mode:play rate:1 time:18.721127818274 duration:252.630204081633 mixer%20volume:50 mixer%20treble:50 mixer%20bass:50 mixer%20pitch:100 playlist%20repeat:2 playlist%20shuffle:0 playlist_cur_index:1 playlist_tracks:3 playlist%20index:1 title:Bounce%20[Original%20Version] playlist%20index:2 title:Open%20Up%20[Radio%20Edit]<LF>"

Subscribe mode example

Request: "a5:41:d2:cd:cd:05 status - 2 subscribe:30<LF>"

Response: "a5:41:d2:cd:cd:05 status - 2 subscribe:30 player_name:127.0.0.1 ... (same as above)

10 seconds later, player is turned off, CLI generates and sends:

"a5:41:d2:cd:cd:05 status - 2 subscribe:30 player_name:127.0.0.1 player_connected:1 power:0<LF>"

30 seconds (the subscribe value) elapse, no changes to the player, the CLI generates and sends:

"a5:41:d2:cd:cd:05 status - 2 subscribe:30 player_name:127.0.0.1 player_connected:1 power:0<LF>"

displaystatus <taggedParameters>

The "displaystatus" query allows subscription to display update events for a player. Details of the latest display change are automatically returned whenever the relevant display update event occurs on that player.

Clients may subscribe to only receive status and warning ('showbriefly') messages, receive normal display updates ('update'), or receive all display updates including menu transitions ('all').

Clients may also subscribe to receive the bitmap which comprises each display update ('bits'). For SB2 and later players, this forwards the display bitmap to the client encoded in base64. Clients may request a reduced width display by setting the width parameter in the status subscription. Note this reduces the width of the display shown on the live player screen as well as for the display forwared to the cli client.

Accepted tagged parameters:

Tag	Description
subscribe	'showbriefly', 'update', 'all' or 'bits' to subscribe and nothing to unsubscribe
width	Reduced width for the display, only used with a subscription for 'bits'

readdirectory <start> <itemsPerResponse> <taggedParameters>

The "readdirectory" query allows to browse filesystems from the server's point of view. This can be used to eg. select music folders. Local filesystems are supported, as are UNC paths on Windows systems (eg. \\server\musicshare).

Please note that on Windows systems the back slashes must either be escaped or replaced by normal slashes. The above path would better be written //server/musicshare.

Accepted tagged parameters:

Tag	Description
folder	the path to the folder to be displayed, eg. c:/music, /Users/mh/Music, //server/share etc.
filter	Filter the output according to one of the following keywords (regulare expressions can be used): • filter:foldersonly - list folders only • filter:filesonly - list files only • filter:musicfiles - list all files considered music files by the server; this is the same filter as is used when scanning the disk for music • filter:filetype:xyz - list file type .xyz only • filter:xyz - any expression filter path/filenames

Returned tagged parameters:

Tag	Description
item	The folder's item: folder, files etc.
isfolder	A flag whether an item is a folder or not.

Example:

> Request: "readdirectory 0 10 root://edelzwerg/mp3"
> Response: "readdirectory 0 10 root%3A%2F%2Fedelzwerg%2Fmp3 count%3A251 item%3A%5C%5Cedelzwerg%5Cmp3%5CAerosmith isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CAir isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CAlanis%20Morissette isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CAldo%20Romano%2C%20Michel%20Benita%2C%20Glenn%20Ferris%2C%20Paolo%20Fresu isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CAli%20Farka%20Toure isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CAll%20Saints isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CManu%20Katche isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CSinead%20O'Connor isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CShakira isdir%3A1 item%3A%5C%5Cedelzwerg%5Cmp3%5CAnastacia isdir%3A1"

Notifications

All commands listed in this document are notifications as well as being commands and can be received when using "listen 1" (if they do not originate from the command-line connection that issued them; no command is echoed twice). Note that queries (for example, "display ? ?") are not notified. Other available notifications are listed below with their meaning. Please note that other notifications or commands may exist, but internal to the server and therefore not documented in this CLI document. Likewise, commands originating from the server may have more parameters than those described in this document, or parameters consisting of internal Perl data structures with strange text representations.

<playerid> client <new|disconnect|reconnect>

A new client is notified using "client new". "client disconnect" is sent when a client disconnects. Unless it reconnects (as signaled by "client reconnect") before a number of minutes, the client will be automatically forgotten by the server (as indicated by command/notification "client forget".)

rescan done

This signals the end of a "rescan" or "wipecache".

library changed <0|1>

This signals the presence or absence of a library has changed.

<playerid> unknownir <ircode> <time>

This signals an IR code unkown by the server. The syntax is the same than the one used by "ir".

Note: This is only available on SB Classic, SB Boom and Transporter. SB Touch and SB Radio handle IR codes locally and do not report them to the server anymore.

<playerid> playlist newsong [<current_title>] [<playlist index>]

This signals the start of a new song, along with its "current_title " and "playlist index". For radio stations, only the "current_title" information is provided.

<playerid> playlist stop

<playerid> playlist pause <0|1>

These signal a change in playing status.

<playerid> prefset [<namespace>] [<prefname>] [<value>]

This signals a preference change.

favorites changed

Sent everytime the favorite database is changed, for any reason, by any process (so the favorites command below will result in this notification being sent).

<playerid> alarm <sound|end|snooze|snooze_end> <id>

"alarm sound" is sent when an alarm sounds; "alarm end", when an alarm ends; "alarm snooze" when an alarm is snoozed; "alarm snooze_end" when a snooze ends (and the alarm resumes). "id" gives the id of the alarm.

Alarm commands and queries

- alarm
- alarms

<playerid> alarm <add|update|delete|enableall|disableall|defaultvolume> <taggedParameters>

The "alarm" command allows to manipulate player alarms.

Accepted tagged parameters:

Tag	Description
id	The id of an existing alarm. This value is mandatory unless you "add" a new alarm.
dow	Day Of Week. 0 is Sunday, 1 is Monday, etc. up to 6 being Saturday. You can define a group of days by concatenating them with "," as separator. Default: 0-6.
dowAdd	Add a single day of the week to the alarm list This takes precendence over anything sent in the dow tag
dowDel	Removes a single day (0-6) of the week from the alarm list This takes precendence over anything sent in the dow tag
enabled	1 if the alarm is enabled. Default: 0.
repeat	1 if the alarm repeats. Default: 1.
time	Time of the alarm, in seconds from midnight. Mandatory when add command is issued
volume	Mixer volume of the alarm. Default: use the default volume for alarms. Mandatory when defaultvolume command is issued
url (or playlisturl)	URL of the alarm playlist. Default: the current playlist. url should be a valid Logitech Media Server audio url. The special value 0 means the current playlist. Example values: • randomplay:track • randomplay:contributor • randomplay:album • randomplay:year See also "favorites items".

Returned tagged parameters:

Tag	Description
id	The id of the newly created or edited alarm.

Examples:

Defining a new alarm

Request: "bd:a5:a9:9b:9d:df alarm add dow:1 enabled:1 playlist:file://some/playlist.m3u time:9000<LF>"

Response: "bd:a5:a9:9b:9d:df alarm add dow:1 enabled:1 playlist:file://some/playlist.m3u time:9000 id:eaf39<LF>"

Deleting an alarm

Request: "bd:a5:a9:9b:9d:df alarm delete id:eaf39<LF>"

Response: "bd:a5:a9:9b:9d:df alarm delete id:eaf39<LF>"

Enabling a previously defined alarm for Mo-Fr

Request: "bd:a5:a9:9b:9d:df alarm update id:eaf39 dow:1,2,3,4,5 enabled:1<LF>"

Response: "bd:a5:a9:9b:9d:df alarm update id:eaf39 dow:1,2,3,4,5 enabled:1 count:1 <LF>"

alarm playlists

The "alarm playlists" returns all the playlists, sounds, favorites etc. available to alarms.

Returned tagged parameters:

Tag	Description
Count	The number of items available
For each playlist:	
title	The item's name or title

category	The category under which the item is grouped (eg. Favorites, Natural Sounds etc.)
url	The item's URL, or the empty value as a placeholder for the current playlist.
singleton	1 if the item is the only one in it's category, or 0 if there's more than one item.

Example:

Request: "alarm playlists 0 3<LF>"

Response: "alarm playlists 0 100 category:The current playlist title:Use Current Playlist url: singleton:1 category:Favorites title:Random%20Artists url:randomplay://contributor singleton:0 category:Favorites title:Random%20Tracks url:randomplay://track singleton:0 count:29 <LF>"

<playerid> alarms <start> <itemsPerResponse> <taggedParameters>

The "alarms" query returns information about player alarms.

Accepted tagged parameters:

Tag	Description
dow	If present, the query returns information about this Day Of Week only. Note this takes precedence over any "filter" parameter. 0 is Sunday, 1 is Monday, etc. up to 6 being Saturday.
filter	One of "all" or "enabled" (the default). To get all possible alarms, use "all". To get only enabled alarms, use "enabled"

Returned tagged parameters:

Tag	Description
fade	1 if the alarms fade in.
count	Number of alarms returned, based on the filters above.

For each alarm:

dow	Days Of Week of this alarm.
enabled	1 if the alarm is enabled.
repeat	1 if the alarm repeats.
time	Time of the alarm, in seconds from midnight.
volume	Mixer volume of the alarm.
url	URL of the alarm playlist, or "CURRENT_PLAYLIST"

Example:

Request: "bd:a5:a9:9b:9d:df alarms 0 3<LF>"

Response: "bd:a5:a9:9b:9d:df alarms 0 3 count:2 fade:0 dow:1 enabled:1 time:3600 volume:50 url:randomplay://track dow:5 enabled:1 time:81000 volume:77 playlist url:file:///Volumes/Smurf/playlists/Playlists/AAA.m3u <LF>"

Additionaly, the following player preferences control the operation of the alarm and can be set/queried using the playerpref command.

alarmfadeseconds (note the case!)

Whether alarms should fade in on this player. Despite the name, this preference is only a boolean and does not control the number of seconds over which alarms fade in. Set to 0 to disable fading; 1 to enable it.

alarmTimeoutSeconds

The number of seconds that an alarm will play for before being automatically stopped. Set to 0 to disable the automatic timeout.

alarmSnoozeSeconds

The number of seconds that a snooze will last for.

alarmsEnabled

Whether any alarm can sound on this player. Set to 0 to prevent any alarm from sounding; 1 to allow them to sound.

alarmDefaultVolume

The volume level (0-100) at which alarms will sound unless they have their own volume specefically set (see "alarm volume").

See also "alarm" under Notifications.

Plugins commands and queries

The following command and queries are proposed by server plugins. The plugin must be enabled in the server configuration for the commands and queries to be available to the CLI client. Please use query "can" to determine if the given command or query is available. Query "radios" can alternatively be used to get a list of available radio station plugin CLI queries.

- radios
- apps
- Live Music Archive, Live365, MP3tunes, Pandora, Podcasts, RadioIO, RadioTime, Rhapsody, Shoutcast, SIRIUS, Staff Picks, RSS
- Favorites
- RandomPlay
- MusicIP

<radios|apps> <start> <itemsPerResponse> <taggedParameters>

The "radios" and "apps" queries return all radio type plugins enabled in the server. These commands enable the client to present a top level "Radio" or "My Apps" respectively menu easily, without hardcoding the plugins nor worrying about them being enabled.

Accepted tagged parameters:

Tag	Description
sort	Field used to sort results, typically "name".

Returned tagged parameters:

Tag	Description
count	Number of results returned by the query.
For each radio plugin:	
cmd	Command, one of "lma", "live365", "mp3tunes", "pandora", "podcast", "radioio", "radiotime", "rhapsodydirect", "shoutcast", "sirius" or "picks", or any other, additionally installed radio or music service type plugin. Item delimiter. This corresponds to the respective queries below.
name	Radio plugin full display name, for example "Live Music Archive".
type	Type of the API of the radio plugin, one of "xmlbrowser" or "xmlbrowser_search". An xmlbrowser_search type must include a search query with the request, for example: "<playerid> search_radio items 0 100 search:KQED"
icon	Relative URL path to an icon for this radio or music service, for example "plugins/Picks/html/images/icon.png"
weight	Numeric weight value to assist in sorting the results. Items should be ordered lowest to highest.

Example:

Request: "radios 0 2<LF>"

Response: "radios 0 2 count:2 cmd:radiotime name:RadioGuide type:xmlbrowser cmd:shoutcast name:SHOUTcast%20Internet%20Radio type:xmlbrowser <LF>"

Live Music Archive, Live365, MP3tunes, Pandora, Podcasts, RadioIO, RadioTime, Rhapsody, Shoutcast, SIRIUS, Staff Picks, RSS

<playerid> <lma|live365|mp3tunes|pandora|podcast|radioio|radiotime|rhapsodydirect|shoutcast|sirius|picks|rss> items <start> <itemsPerResponse> <taggedParameters>

These instructions are valid for all XMLBrowser based plugins. While the CLI queries are identical, returned values may vary between plugins. All queries to fetch data from the Internet, and consequently, as experienced in the web interface, there may be a significant delay before these queries return data. Data is cached however so subsequent queries performed immediately return much faster.

Please check at the end of this section for specific commands for some radios.

Accepted tagged parameters:

Tag	Description
item_id	The id of an item to be returned. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3
search	When a list of items is to be returned, it can be filtered by it's name or title.
want_url	If set to 1, urls are returned by the query, otherwise they aren't.

Returned tagged parameters:

Tag	Description
networkerror	Returned with value 1 if there was a network error accessing the content source.
count	The number of items available at the selected level.

For each element:

id	An item's hierarchical id. Item delimiter.
name	An item's (station, track, podcast etc.) name or title.
hasitems	Whether or not an item has sub-items. May indicate the number of sub-items.
isaudio	Whether or not an item is audio, determined from the other fields (type, etc.)
type	Stream content type. Value "link" means sub-items must be fetched. Example types: link, text, audio, playlist.
url	URL of the station or track (only returned if parameter "want_url" is set to 1). Although, the station can be played using the "playlist play" command, an equivalent command that operates on the id is provided below.
description	Some streams (podcasts) provide further information than the title.
length	Some streams (podcasts) contain information about their size/length.

Example:

> Request: "picks items 0 2 item_id:0<LF>"
>
> Response: "picks items 0 2 item_id:0 count:19 id:0.0 name:Alternative | RADIO%20DAVIDBYRNE.COM hasitems:0 id:0.1 name:Alternative | KCRW Music hasitems:0 <LF>"

<playerid> <lma|live365|mp3tunes|pandora|podcast|radioio|radiotime|rhapsodydirect|shoutcast|picks|rss> playlist <play|load|insert|add> <taggedParameters>

This command adds or plays an item. Though playback of these stations could be achieved by the standard playlist methods, calling this command will ensure that the station's name is displayed, not only it's url (if possible).

If item_id defines an item that can't be played, but contains playable subitems, then these will be played instead. This allows to eg. play all tracks of a genre.

Accepted tagged parameters:

Tag	Description
item_id	The id of an item to be played. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3

Example:

> Request: "6e:ef:54:e9:02:b0 shoutcast playlist play item_id:1.1<LF>"
>
> Response: "6e:ef:54:e9:02:b0 shoutcast playlist play item_id:1.1<LF>"

<playerid> <pandora> rate <0|1>

Enables rating the Pandora current song. If rated 0, the song is skipped if possible.

Example:

Request: "6e:ef:54:e9:02:b0 pandora rate 0<LF>"

Response: "6e:ef:54:e9:02:b0 pandora rate 0<LF>"

<playerid> <pandora> skipTrack

Skips the currently playing Pandora song.

Example:

Request: "6e:ef:54:e9:02:b0 pandora skipTrack<LF>"

Response: "6e:ef:54:e9:02:b0 pandora skipTrack<LF>"

Favorites

favorites items <start> <itemsPerResponse> <taggedParameters>

The "favorites items" query returns all server favorites. This query is based on XMLBrowser.

Accepted tagged parameters:

Tag	Description
item_id	The id of a favorite to be returned. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3
search	When a list of items is to be returned, it can be filtered by it's name or title.
want_url	If set to 1, urls are returned by the query, otherwise they aren't.

Returned tagged parameters:

Tag	Description
count	The number of items available at the selected level.
For each element:	
id	An item's hierarchical id. Item delimiter.
name	An item's (favorite or folder) name.
hasitems	Whether or not an item has sub-items. May indicate the number of sub-items.
url	URL of the station or track (only returned if parameter "want_url" is set to 1). Although, the station can be played using the "playlist play" command, an equivalent command that operates on the id is provided below.

Example: See other XMLBrowser based queries above.

favorites exists <id | url>

The "favorites exists" command is used to check whether a given track ID or URL exists in favorites.

Returned tagged parameters:

Tag	Description
exists	Returned with value 1 if the ID or URL exists in favorites.
index	If exists is 1, the index of the ID or URL in favorites.

Example:

Request: "favorites exists file:///... <LF>"

Response: "favorites exists file:///... exists:1 index:5<LF>"

favorites add <taggedParameters>

The "favorites add" command adds a favorite.

Accepted tagged parameters:

Tag	Description
item_id	The id of a favorite to be inserted. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3. Room is made to accommodate the new favorite. If no item_id is provided, the favorite is added at position 0.
title	Favorite title (mandatory)
url	Favorite url (mandatory)
hotkey	Optional hotkey to associate with the Favorite

Returned tagged parameters:

Tag	Description
count	Returned with value 1 if adding the favorite was successful.

Example:

Request: "favorites add url:file:///... title:BestSong<LF>"

Response: "favorites add url:file:///... title:BestSong count:1<LF>"

favorites addlevel <taggedParameters>

The "favorites addlevel" command adds a favorite level (a folder).

Accepted tagged parameters:

Tag	Description
item_id	The id of a level to be inserted. The id represents the hierarchical structure of the level using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3. Room is made to accommodate the new level. If no item_id is provided, the level is added at position 0.
title	Level title (mandatory)

Returned tagged parameters:

Tag	Description
count	Returned with value 1 if adding the level was successful.

Example:

Request: "favorites addlevel title:Favourites<LF>"

Response: "favorites addlevel title:Favourites count:1<LF>"

favorites delete <taggedParameters>

The "favorites delete" command deletes a favorite or a level.

Accepted tagged parameters:

Tag	Description
item_id	The id of a favorite or level to be deleted. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3. This parameter is mandatory.

Example:

Request: "favorites delete item_id:1.2.3.4.5<LF>"

Response: "favorites delete item_id:1.2.3.4.5<LF>"

favorites rename <taggedParameters>

The "favorites rename" command renames a favorite or a level.

Accepted tagged parameters:

Tag	Description
item_id	The id of a favorite or level to be renamed. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3. This parameter is mandatory.
title	The new title to rename this item to. This parameter is mandatory.

Example:

Request: "favorites rename item_id:1.2.3.4.5 title:NewTitle<LF>"

Response: "favorites rename item_id:1.2.3.4.5 title:NewTitle<LF>"

favorites move <taggedParameters>

The "favorites move" command moves a favorite or a level.

Accepted tagged parameters:

Tag	Description
from_id	The id of a favorite or level to be moved. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3. This parameter is mandatory.
to_id	The id to move the favorite or level to. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3. This parameter is mandatory.

Example:

Request: "favorites move from_id:1.2.3.4.5 to_id:5.4.3.2.1<LF>"

Response: "favorites move from_id:1.2.3.4.5 to_id:5.4.3.2.1<LF>"

<playerid> <favorites> playlist <play|load|insert|add> <taggedParameters>

This command adds or plays a favorite.

If item_id defines an item that can't be played, but contains playable subitems, then these will be played instead. This allows to eg. play all tracks of a genre.

Accepted tagged parameters:

Tag	Description
item_id	The id of an item to be played. The id represents the hierarchical structure of the file using a dotted syntax similar to the one used in SNMP, like eg. 2.0.9.3

Example:

Request: "6e:ef:54:e9:02:b0 favorites playlist play item_id:1.1<LF>"

Response: "6e:ef:54:e9:02:b0 favorites playlist play item_id:1.1<LF>"

RandomPlay

<playerid> randomplay <tracks|albums|contributors|year>

The "randomplay" command starts a random mix of the given type.

Example:

Request: "04:20:00:12:23:45 randomplay albums<LF>"

Response: "04:20:00:12:23:45 randomplay albums<LF>"

<playerid> randomplaygenrelist

This returns a formatted list of genres which is for use on the Jive platform

Example:

Request: "04:20:00:12:23:45 randomplaygenrelist<LF>"

Response: "04:20:00:12:23:45 randomplaygenrelist count%3A103 offset%3A0 actions%3AHASH(0xb804350) checkbox%3A1 text%3A80s actions%3AHASH(0xb8042a8) checkbox%3A1 text%3AAcid%20Jazz actions%3AHASH(0xb80438c) checkbox%3A1 text%3AAcoustic actions%3AHASH(0xb804494) checkbox%3A0 text%3AAdvertisement actions%3AHASH(0xb8cdcc0) checkbox%3A1 text%3AAfropop actions%3AHASH(0xb8cdf9c) checkbox%3A1 <LF>"

<playerid> randomplaychoosegenre <0|1>

Turn a particular genre on/off in random mix

Example:

Request: "04:20:00:12:23:45 randomplaychoosegenre Afropop 1<LF>"

Response: "04:20:00:12:23:45 randomplaychoosegenre Afropop 1<LF>"

<playerid> randomplaygenreselectall <0|1>

Turn all genres on/off in random mix

Example:

Request: "04:20:00:12:23:45 randomplaygenreselectall 1<LF>"

Response: "04:20:00:12:23:45 randomplaychoosegenre 1<LF>"

MusicIP

<playerid> musicip mix <taggedParameters>

The "musicip mix" query creates a MusicIP mix on the given additional Parameters.

Accepted tagged parameters:

Tag	Description
mood	A MusicIP mood's name.
artist_id	Artist ID, to create a mix based on that artist.
album_id	Album ID, to create a mix based that album.
song_id	Song ID, to reate a mix based track.
genre_id	Genre ID, to create a mix based that genre.
year	Year, to create a mix based on that year.
tags	Determines which tags are returned. Each returned tag is identified by a letter (see command "songinfo" for a list of possible fields and their identifying letter). The default tags value for this command is "gald".

Example:

Request: "04:20:00:12:23:45 musicip album_id:23<LF>"

Response: "04:20:00:12:23:45 musicip album_id:23 id:2441 title:I'll%20Be%20Back artist:The%20Beatles album:A%20Hard%20Day's%20Night id:1807 title:Ni%20Remords%2C%20Ni%20Regrets artist:Stephan%20Eicher album:Non%20Ci%20Badar%2C%20Guarda%20E%20Passa... id:2506 title:Fire artist:Jimi%20Hendrix album:Are%20You%20Experienced ... <LF>"

<playerid> musicip <play|add>

Play or add the last MusicIP mix.

Example:

Request: "04:20:00:12:23:45 musicip play<LF>"

Response: "04:20:00:12:23:45 musicip play<LF>"

<playerid> musicip moods

Return list of available moods

Example:

Request: "04:20:00:12:23:45 musicip moods<LF>"

Response: "04:20:00:12:23:45 musicip moods name:Soft name:Guitars count:2<LF>"

Deprecated commands and queries

The following commands are still supported but their usage is not recommended:

- mode
- playlisttracks

<playerid> mode <play|pause|stop>

The "mode" command allows to set the player playlist mode directly. **It is deprecated**. "mode play" has been replaced by "play" "mode stop" by "stop" and "mode pause" by "pause 1".

playlisttracks <start> <itemsPerResponse> <taggedParameters>

The "playlisttracks" command returns the tracks of a given playlist, **is deprecated**, and has been replaced by the "playlists tracks" query.