LODE®
MULTI ROOM AUDIO SOLUTIONS

Lode Audio LS1 API Specification V 1.4

Revision History

Date	Version	Change
1st Just 13	1.2	Added play line in command.
30th Sep 13	1.3	Added add to favourite and delete favourite commands
24th Mar 14	1.4	Added movement of tracks in queue and fixed volume for player.

Connectivity

The LS1 API has two modes of connectivity. A telnet interface and a raw socket interface. Either of these can be used and the following sections apply to both protocols. If Telnet is chosen, consider the additional control signals required by the telnet specification.

Testing

Immediate testing can be performed via both protocols on a Unix system by telneting into either protocol's port and issuing commands. For example:

```
> Telnet 192.168.20.10 6667
Trying 192.168.20.10...
Connected to 192.168.20.10.
Escape character is '^]'.
?ZONES
~ZONES,{Study},{Living Room}
```

Snooping

During the development process, it is advantageous to see the traffic passing between the control system and LS1. To this end, a snooping shell is provided on port 7000. To access the LS1 Snoop Shell, telnet to port 7000, no authentication necessary. From here you can list the currently active sessions and attach to them to see all the traffic being passed in both directions. For details of how to achieve this, type 'help' at the command prompt, having established a telnet session.

Conventions

Action commands are prefixed by the '#' character. Query commands are prefixed by the '?' character. All responses are asynchronous and prefixed by the '~' character.

In many cases, the API will return multidimensional data. The convention used is to enclose each row with curly brackets: '{' and '}'. The content of the row will be comma delimited as usual.

As the API deals with parameters that may themselves contain a comma, and the API parameter delimiter is itself a comma, these fields will be double quoted at each end to indicate the beginning and end of a parameter, to avoid a conflict. This allows for fields to contain the comma delimiter character. For example ""Album name, with a comma in"".

Players Summary

Query to get a list of the zone player names. Note that the ~PLAYERS response may be issued asynchronously in the event of a topology change.

This query may be useful to initialise the state of a client application, where a list of the zone player's names is required.

Players Query Format

```
Query
|
?PLAYERS
 |
 Command
```

Example Players Query

Operation	Command String
Execute ?PLAYERS	
Query the player names	?PLAYERS
Response ~PLAYERS	
List of player names	~PLAYERS,<player name>,<player name>

Zones Grouping Summary

These commands are intended to control how the player's are grouped together. The API provides commands to query the current zoning structure as well as manipulate the zoning structure by adding and removing members.

The API will asynchronously send a zone response in the event of any change to the zoning structure. The zone response details the structure of the zone grouping as a two dimensional list. The first member of each list is the controller for that zone, followed by a list of member players contained within that zone.

Zones Query Format

The API will respond to the zones query with a snapshot of the current zoning structure. This is useful to initialise client structures.

```
Query
|
?ZONES
 |
 Command
```

Example Zones Query

Operation	Command String
Execute ?ZONES	
Query the room grouping	?ZONES
Response ~ZONES	
Room Grouping	~ZONES, {<Controller player name>,<Member player name>, <Member player name>}, {<Controller player name>,<Member player name>, <Member player name>} Note. The response is a two dimensional array of groups. The first element of each dimension is the controller followed by a list of the player members.

Add Member Format

An action command used to add a player to a group

```
Action
|
#ADDMEMBER,<controller/player>,<player to add>
 |            |                    |
 Command      Parameter               Parameter
```

Example Add Member Action

Operation	Command String
Execute #ADDMEMBER	
Add the Study player to the Lounge group	#ADDMEMBER,Lounge,Study
Response ~ZONES	
Room Grouping	~ZONES,{Lounge, Study},{Bedroom}

Remove Member Format

An action command used to remove a player from a group

```
Action
|
#REMOVEMEMBER,<player to Remove>
 |              |
 Command        Parameter
```

Example Remove Member Action

Operation	Command String
Execute #REMOVEMEMBER	
Remove the Study player from any group membership.	#REMOVEMEMBER,Study
Response ~ZONES	
Room Grouping	~ZONES,{Lounge},{Study},{Bedroom}

Party Mode Action

The party mode action is a convenience action that will group all players into a single group.

```
Action
|
#PARTYMODE,<current player>
 |           |
 Command     Parameter
```

Example Party Mode Action

Operation	Command String
Execute #PARTYMODE	
Start a party…	#PARTYMODE,Lounge
Response ~ZONES	
Room Grouping	~ZONES,{Lounge,Study,Bedroom}

Transport Controls Summary

A set of commands to control a player's audio transport.

Volume Query Format

Query the volume of a player.

```
Operation
|
?VOLUME,<Player Name>
 |
 Command
```

Example Volume Query Commands

Operation	Command String
Execute ?VOLUME	
Query the player's volume	?VOLUME,<player name>
Response ~VOLUME	
Player volume	~VOLUME,<player name>,<0 to 100 volume level>

Note. Players with a fixed volume will return -1

Volume Action Format

Set the volume of a player.

```
Operation
|
#VOLUME,<Player Name>,<Volume (0 to 100)>
 |
 Command
```

Example Volume Action Commands

Operation	Command String
Execute #VOLUME	
Set player's volume	#VOLUME,<player name>, <0 to 100 volume level>
Response ~VOLUME	
Player volume	~VOLUME,<player name>,<0 to 100 volume level>

Balance Query Format

Query the balance of a player.

```
Operation
|
?BALANCE,<Player Name>
 |
 Command
```

Example Balance Query Commands

Operation	Command String
Execute ?BALANCE	
Query the player's balance	?BALANCE,<player name>
Response ~BALANCE	
Player balance	~BALANCE,<player name>,<-100 ~0 ~ 100 balance level>

Balance Action Format

Set the balance of a player.

```
Operation
|
#BALANCE,<Player Name>,<Balance (-100 to 0 to 100)>
 |
 Command
```

Example Balance Action Commands

Operation	Command String
Execute #BALANCE	
Set player's balance	#BALANCE,<player name>, <-100 to 0 to 100 balance level>
Response ~BALANCE	
Player balance	~BALANCE,<player name>,<-100 to 0 to 100 balance level>

Bass Query Format

Query the bass of a player.

```
Operation
|
?BASS,<Player Name>,
 |
 Command
```

Example Bass Query Commands

Operation	Command String
Execute ?BASS	
Query the player's bass	?BASS,<player name>
Response ~BASS	
Player bass	~BASS,<player name>,<-100 to 0 to 100 bass level>

Bass Action Format

Set the bass of a player.

```
Operation
|
#BASS,<Player Name>,<Bass (-100 to 0 to 100)>
 |
 Command
```

Example Bass Action Commands

Operation	Command String
Execute #BASS	
Set player's bass	#BASS,<player name>, <-100 to 0 to 100 bass level>
Response ~BASS	
Player bass	~BASS,<player name>,<-100 to 0 to 100 bass level>

Treble Query Format

Query the Treble of a player.

```
Operation
|
?TREBLE,<Player Name>
 |
 Command
```

Example Treble Query Commands

Operation	Command String
Execute ?TREBLE	
Query the player's Treble	?TREBLE,<player name>
Response ~TREBLE	
Player Treble	~TREBLE,<player name>,<-100 to 0 to 100 Treble level>

Treble Action Format

Set the Treble of a player.

```
Operation
|
#TREBLE,<Player Name>,<Treble (-100 to 0 to 100)>
 |
 Command
```

Example Treble Action Commands

Operation	Command String
Execute #TREBLE	
Set player's Treble	#TREBLE,<player name>, <-100 to 0 to 100 Treble level>
Response ~TREBLE	
Player Treble	~TREBLE,<player name>,<-100 to 0 to 100 Treble level>

Loudness Action Format

Turn on/off the loudness of a player.

```
Operation
|
#LOUDNESS,<Player Name>,[‘ON’/’1’/true or ‘OFF’/’0’/false]
 |
 Command
```

Example Loudness Action

Operation	Command String
Execute #LOUDNESS	
Loudness the Study player.	#LOUDNESS,Study,ON
Response ~LOUDNESS	
Player Loudness state	~LOUDNESS,Study,1

Loudness Query

Query the Loudness state of a player.

```
Operation
|
?LOUDNESS,<Player Name>
 |
 Command
```

Example Loudness Query

Operation	Command String
Execute #LOUDNESS	
Loudness of Study?	?LOUDNESS,Study
Response ~LOUDNESS	
Player Loudness state	~LOUDNESS,Study,1

Mute Action Format

Mute/unmute a player

```
Operation
|
#MUTE,<Player Name>,[‘ON’/’1’ or ‘OFF’/’0’]
 |
 Command
```

Example Mute Action

Operation	Command String
Execute #MUTE	
Mute the Study player.	#MUTE,Study,ON
Response ~MUTE	
Player mute state	~MUTE,Study,1

Mute Query

Query the mute state of a player.

```
Operation
|
?MUTE,<Player Name>
 |
 Command
```

Example Mute Query

Operation	Command String
Execute #MUTE	
Mute of Study?	?MUTE,Study
Response ~MUTE	
Player mute state	~MUTE,Study,1

Pause Action

Pause a player.

```
Operation
|
#PAUSE,<Player Name>
 |
 Command
```

Example Pause Command

Operation	Command String
Execute #PAUSE	
Pause Study	#PAUSE,Study
Response ~TRANSPORT	
Player transport state	~TRANSPORT,Study,PAUSED_PLAYBACK

Play Action

Play a player.

```
Operation
|
#PLAY,<Player Name>
 |
 Command
```

Example Play Command

Operation	Command String
Execute #PLAY	
Play Study	#PLAY,Study
Response ~TRANSPORT	
Player transport state	~TRANSPORT,Study,PLAYING

Previous Action

Play the previous track in the queue.

```
Operation
|
#PREVIOUS,<Player Name>
 |
 Command
```

Example Previous Command

Operation	Command String
Execute #PREVIOUS	
Play previous track	#PREVIOUS,Study
Response ,	
Player track	~TRACK,Study,""French Cuisine"",""Alif Tree"",""Deadly Species"",http://192.168.20.10:8080/img=-1161822296.png,1,10,221
Response ~NEXTTRACK	
Player's next track	~NEXTTRACK,Study,Belle

Next Action

Play the next track in the queue.

```
Operation
|
#NEXT,<Player Name>
 |
 Command
```

Example Next Command

Operation	Command String
Execute #NEXT	
Play next track	#NEXT,Study
Response ~TRACK	
Player track	~TRACK,Study,""French Cuisine"",""Alif Tree"",""Belle"",http://192.168.20.10:8080/img=1052897533.png,2,10,221
Response ~NEXTTRACK	
Player's next track	~NEXTTRACK,Study,Enough

Transport Query

Query a player’s transport state.

```
Operation
|
?TRANSPORT,<Player Name>
 |
 Command
```

Example Play Command

Operation	Command String
Execute ?TRANSPORT	
Query Study player’s transport	?TRANSPORT,Study
Response ~TRANSPORT	
Player transport state	~TRANSPORT,Study,PLAYING

Transport State Response

The transport state response is issued whenever a player is paused or played.

```
Operation
|
~TRANSPORT,<Player Name>,[see state table]
 |
 Command
```

Transport State Table

Description	Value
The specified player is playing	PLAYING
The specified player is paused.	PAUSED_PLAYBACK
The specified player has been stopped. Issued when streaming audio is being directly played, e.g. Radio.	STOPPED

Track Query

Query a player's current track.

```
Operation
|
?TRACK,<Player Name>
 |
 Command
```

Example Track Query

Operation	Command String
Execute ?TRACK	
Query Study player's track	?TRACK,Study
Response ~TRACK	
Player transport state	~TRACK,Study,""French Cuisine"",""Alif Tree"",""Belle"",http://192.168.20.10:8080/img=1052897533.png,2,10,221

Track Response

Query a player's current track.

```
Operation
|
?TRACK,[see response parameter table]
 |
 Command
```

Track Response Parameter Table

Parameter	Description
1	Player
2	“” Album “” (Double quoted in case of special characters)
3	“” Artist “”(Double quoted in case of special characters)
4	“” Title “”(Double quoted in case of special characters)
5	Album Art URI
6	Current track no.
7	Number of tracks.
8	Track duration (seconds)

```
Note. The LS1 unit will cache and resize the album art exposing a tiny URL in the
response.
```

Track Progress Response

If configured to do so, an asynchronous track progress response message is sent during the playback of a track. This is configured through the web console and can be enabled or disabled as desired. Track progress updates can be configured to be sent at 1, 2, 5, and 10 second intervals.

```
Operation
|
~TRACKPROGRESS,[see response parameter table]
 |
 Command
```

Track Progress Response Parameter Table

Parameter	Description
1	Player
2	Duration (seconds)
3	Offset (seconds)
4	% Offset
5	Offset String “MM:SS”
6	Duration String “MM:SS”
7	Remaining String “MM:SS”

```
Note. Track progress updates will cause increased load to legacy control systems due
to the volume communication with the LS1 unit.
```

Seek Action

Jump to a particular section of a track.

```
Operation
|
#SEEK,<Player Name>,<relative>,<numerator>
 |
 Command
```

This command will seek to an arbitrary relative point in the track. For example, to jump to a relative percentage, the numerator would be 100. For finer grained control increase the numerator.

Example Seek Action

Operation	Command String
Execute #SEEK	
Seek player to 10% of track.	#SEEK,Study,10,100
Response ~TRACKPROGRESS	
Player track progress	~TRACKPROGRESS,...

Fast-Forward Action

Fast-forward a track.

```
Operation
|
#FASTFORWARD,<Player Name>,<seconds to fast forward>
 |
 Command
```

Example Seek Action

Operation	Command String
Execute #FASTFORWARD	
Fast-forward 5 seconds	#FASTFORWARD,Study,5
Response ~TRACKPROGRESS	
Player track progress	~TRACKPROGRESS,...

Rewind Action

Rewind a track.

```
Operation
|
#REWIND,<Player Name>,<seconds to fast forward>
 |
 Command
```

Example Seek Action

Operation	Command String
Execute #REWIND	
Rewind 5 seconds	#REWIND,Study,5
Response ~TRACKPROGRESS	
Player track progress	~TRACKPROGRESS,...

Set Play Mode Action

Sets the play mode for a particular player.

```
Operation
|
#PLAYMODE,<Player Name>,[see mode table]
 |
 Command
```

Play Mode Table

Value	Description
NORMAL	Normal play mode. One track after another, no repeat.
REPEAT_ALL	Will repeat all tracks in the queue
SHUFFLE	Randomly play tracks from the queue.
SHUFFLE_NOREPEAT	As shuffle but will not repeat tracks.

Example Play Mode Action

Operation	Command String
Execute #PLAYMODE	
Set Study player to repeat all.	#SETPLAYMODE,Study,REPEAT_ALL
Response ~PLAYMODE	
Player transport state	~PLAYMODE,Study,REPEAT_ALL

Play Mode Response

Response indicating a players play mode.

```
Operation
|
~PLAYMODE,<Player Name>,[see mode table]
 |
 Command
```

Play Mode Query

The play mode may be queried at any time with the following command. This may be useful for initialisation purposes, where the client application needs to establish the state of the play mode.

```
Operation
|
?PLAYMODE,<Player Name>
 |
 Command
```

Next Track Query

Query a player's next track.

```
Operation
|
?NEXTTRACK,<Player Name>
 |
 Command
```

Example Play Command

Operation	Command String
Execute ?NEXTTRACK	
Query Study player's next track	?NEXTTRACK,Study
Response ~NEXTTRACK	
Player's next track	~NEXTTRACK,Study,My Soul

Next Track Response

Response indicating a player's next track.

```
Operation
|
~NEXTTRACK,<Player Name>,""<track>""
 |
 Command
```

The track name will be double quoted in case of special characters.

Browse Summary

Browse Action

Browse the media library and service content.

```
Operation
|
#BROWSE,<Player Name>,””<ID>””,<index>,<count>
 |
 Command
```

```
Note. the ID is double quoted.
```

Example Next Command

Operation	Command String
Execute #NEXT	
Play next track	#BROWSE,Study,””””,0,10
Response ~BROWSE	
Root content list	~BROWSE,””””,6,6, {""A:"",""Music Library"","""",null,true,””””}, {""S:"",""Folders"","""",null,true,””””}, {""SQ:"",""Playlists"","""",null,true,””””}, {""AI:"",""Line-In"","""",null,true,””””}, {""Service:9::root"",""Spotify"","""",null,true,””””}, {""Service:254::root"",""TuneIn"","""",null,true,””””}

Browse Response

Browse response to the browse action.

```
Operation
|
~BROWSE,<parent ID>,<total matches>,<number returned>,
            { see entry table },
            { see entry table },
            { see entry table }…
 |
 Command
```

Browse Entry Table

Parameter	Description
1	“” <ID> “”(Double quoted in case of special characters)
2	“” <title> “”(Double quoted in case of special characters)
3	“” <artist> “”(Double quoted in case of special characters)
4	Album Art URI
5	Item Attributes – See Attribute table
6	“” Resource URI “”

Attribute Entry Table

Attribute	Description
CONTAINER	Can navigate into this item.
PLAYABLE	Can play this item.
QUEUEABLE	Can add this item to the play queue.
PLAYLIST	Item is a playlist and can be renamed and deleted.

Play Now Action

Instruct a player to play the URI immediately.

```
Operation
|
#PLAYNOW,<Player Name>,""<Resource URI>""
 |
 Command
```

```
Note. It is important to double quote the resource URI as indicated above. This will
ensure that if the URI contains commas, these are not interpreted as delimiters.
```

Example Play Now Action

Operation	Command String
Execute #PLAYNOW	
Play track in the Study	#PLAYNOW,Study,""x-file-cifs://mac/itunes/Alif%20Tree/French%20Cuisine/02%20Belle%201.mp3""
Response ~QUEUECHANGED	
	~QUEUECHANGED,Study,0
Response ~TRACK	
	~TRACK,Study,""French Cuisine"",""Alif Tree"",""Belle"",http://192.168.20.10:8080/img=1052897533.png,1,1,221
Response ~NEXTTRACK	
	~NEXTTRACK,Study,

Play Next Action

Instruct a player to play the URI next.

```
Operation
|
#PLAYNEXT,<Player Name>,""<Resource URI>""
 |
 Command
```

```
Note. It is important to double quote the resource URI as indicated above. This will
ensure that if the URI contains commas, these are not interpreted as delimiters.

Note. This operation is not valid for the TuneIn service.
```

Play Line In Action

Instruct a player to play the line in called <name>.

```
Operation
|
#PLAYLINEIN,<Player Name>,""<Line In Name>""
 |
 Command
```

Add to Favourite Action

Add an item to the favourites list

```
Operation
|
#ADDTOFAV,<Player Name>,""<Resource URI>"",""<Parent Resource URI>""
 |
 Command
```

```
Note. It is important to double quote the resource URI as indicated above. This will
ensure that if the URI contains commas, these are not interpreted as delimiters.

Note. This operation is not valid for all browse entries, only ones with attribute
FAVOURITEBALE
```

Delete from Favourite Action

Add an item to the favourites list

```
Operation
|
#DELETEFAV,<Player Name>,""<Resource URI>""
 |
 Command
```

```
Note. It is important to double quote the resource URI as indicated above. This will
ensure that if the URI contains commas, these are not interpreted as delimiters.
```

Add to Queue Action

Instruct a player to play the URI next.

```
Operation
|
#ADDTOQUEUE,<Player Name>,""<Resource URI>""
 |
 Command
```

```
Note. It is important to double quote the resource URI as indicated above. This will
ensure that if the URI contains commas, these are not interpreted as delimiters.

Note. This operation is not valid for streaming services such as TuneIn.
```

Replace Queue Action

Instruct a player to play the URI next.

```
Operation
|
#REPLACEQUEUE,<Player Name>,””<Resource URI>””
 |
 Command
```

```
Note. It is important to double quote the resource URI as indicated above. This will
ensure that if the URI contains commas, these are not interpreted as delimiters.

Note. This operation is not valid for streaming services such as TuneIn.
```

Search Criteria Query

In order to conduct a valid search on the content, the criteria must first be queried. The response defines the parameters of any subsequent searches.

```
Operation
|
?SEARCHCRITERIA
 |
 Command
```

Example Search Criteria Query

Operation	Command String
Execute ?SEARCHCRITERIA	
	?SEARCHCRITERIA
Response ~SEARCHCRITERIA	
	~SEARCHCRITERIA,A:,Music Library,{A:ALBUM:,Album,A:ALBUMARTIST:,Artist,A:TRACKS:,Tracks},Service:9::,Spotify,{salb,Album,sart,Artist,strk,Track},Service:254::,TuneIn,{search:station,Station,search:show,Show,search:host,Host}

Search Criteria Response

The search criteria response details an item list of content items that may be searched. At the end of each of the searchable content items is another list containing valid search criteria for that item.

```
Operation
|
~SEARCHCRITERIA, [<ID>,<title>,{see criteria table}]…
 |
 Command
```

Search Criteria Table

Parameter	Description
1	<Criteria ID>
2	<criteria title>

Search Criteria Example

```
~SEARCHCRITERIA,
        A:,Music Library,
                {A:ALBUM:,Album,A:ALBUMARTIST:,Artist,A:TRACKS:,Tracks},
        Service:9::,Spotify,
                {salb,Album,sart,Artist,strk,Track},
        Service:254::,TuneIn,
                {search:station,Station,search:show,Show,search:host,Host}
```

In this example, there are three possible content items that can be searched:

- Music Library
- Spotify
- TuneIn

The Music library may be searched with the following criteria:

- Album
- Artist
- Tracks

TuneIn may be searched with the following criteria:

- Station
- Show
- Host

Spotify with more or less the same criteria as the Music Library, but with different IDs

Search Action

Search a content item against particular criteria

```
Operation
|
#SEARCH,<Player Name>,<Content ID>,<Criteria ID>,
            <search term>,<index>,<count>
 |
 Command
```

Example Search Action

Operation	Command String
Execute #SEARCH	
Search the music library for an artist called 'freq'.	#SEARCH,Study,A:,A:ALBUMARTIST:,freq,0,10
Response ~BROWSE	
	~BROWSE,1,1,{""A:ALBUMARTIST/Freq%20Nasty"","" Freq Nasty"","""",,true,””x-rincon-playlist:RINCON_000E58A18FA001400#A:ALBUMARTIST/Freq%20Nasty””}

Delete Playlist Action

Delete the selected playlist

```
Operation
|
#DELETEPLAYLIST,<Player Name>,<Content ID>
 |
 Command
```

Note. It may be advisable to re-issue a browse command after performing the delete playlist action to see the results.

Example Delete Playlist Action

Operation	Command String
Execute #DELETEPLAYLIST	
Delete the playlist	#DELETEPLAYLIST,Study,SQ:3

Rename Playlist Action

Rename the selected playlist

```
Operation
|
#RENAMEPLAYLIST,<Player Name>,<Content ID>,<old name>,<new name>
 |
 Command
```

Note. It may be advisable to re-issue a browse command after performing the rename playlist action to see the results.

Example Rename playlist Action

Operation	Command String
Execute #RENAMEPLAYLIST	
Delete the playlist	#RENAMEPLAYLIST,Study,SQ:3,""Bad name"",""Good""

Queue Summary

Queue Query

Query the play queue.

```
Operation
|
?QUEUE,<Player Name>,<index>,<count>
 |
 Command
```

```
Note. the ID is double quoted.
```

Example Queue Query

Operation	Command String
Execute ?QUEUE	
Request the play queue	?QUEUE,Study,0,6
Response ~QUEUE	
	~QUEUE,Study,12, {Q:0/1,""Sub-Concious"","""", http://192.168.20.10:8080/img=-795754948.png }, {Q:0/2,""Boomin' Back Atcha"",""Freq Nasty/Phoebe One"", http://192.168.20.10:8080/img=519208162.png }, {Q:0/3,""Freq-A-Zoid"",""Freq Nasty"", http://192.168.20.10:8080/img=801076872.png }, {Q:0/4,""Se15"",""Freq Nasty"", http://192.168.20.10:8080/img=359364840.png }, {Q:0/5,""Revolution Inc"",""Akure Wall/Freq Nasty"", http://192.168.20.10:8080/img=-896500110.png }, {Q:0/6,""Mindsweeper"",""Freq Nasty"", http://192.168.20.10:8080/img=2035921686.png }

Queue Response

The queue response indicated the contents of the queue from the supplied index and count by issuing the ?QUEUE query. It is not supplied asynchronously, rather a ~QUEUECHANGED event is issued.

```
Operation
|
~QUEUE,<player>,<total queue size>,
                        [<ID>,<title>,{see criteria table}]…
 |
 Command
```

Queue Response Table

Parameter	Description
1	<Queue item ID>
2	<title>
3	<artist>
4	<album art>

Queue Changed Response

The 'queue changed' event is issued whenever a player's queue is changed. This is the event to re-request the visible portion of the queue being displayed.

```
Operation
|
~QUEUECHANGED,<player>,<total queue size>
 |
 Command
```

Clear Queue Action

Action to remove all the items from the player's play queue. Any currently playing tracks will be stopped.

```
Operation
|
#CLEARQUEUE,<player>
 |
 Command
```

Example Clear Queue Command

Operation	Command String
Execute #CLEARQUEUE	
	#CLEARQUEUE,Study
Response ~QUEUECHANGED	
	~QUEUECHANGED,Study,0

Play Queue Item Action

This command will instruct a player to start playing a particular track contained within the play queue.

```
Operation
|
#PLAYQUEUE,<player>,<item no>
 |
 Command
```

Example Play Queue Command

Operation	Command String
Execute #PLAYQUEUE	
	#PLAYQUEUE,Study,3
Response ~QUEUECHANGED	
	~QUEUECHANGED,Study,0

Remove From Queue Action

Remove an item from the player's queue.

```
Operation
|
#REMOVEFROMQUEUE,<player>,<item no>
 |
 Command
```

Example Remove From Queue Command

Operation	Command String
Execute #REMOVEFROMQUEUE	
	#REMOVEFROMQUEUE,Study,3
Response ~QUEUECHANGED	
	~QUEUECHANGED,Study,12

## RE-ORDER TRACK IN QUEUE

Move and item in the queue from one position to another.

```
Operation
|
# REORDERTRACKINQUEUE,<player>,<item no>,<destination position>
 |
 Command
```

Example Re-Order In Queue Command

Operation	Command String
Execute #REORDERTRACKINQUEUE	
	# REORDERTRACKINQUEUE,Study,3,8
Response ~QUEUE	
	See Queue Response

Save Queue Action

Will save the contents if there current player's queue into the Playlists folder.

```
Operation
|
#SAVEQUEUE,<player>,<name of saved queue>
 |
 Command
```

Example Save Queue Command

Operation	Command String
Execute #SAVEQUEUE	
	#SAVEQUEUE,Study,Test
Response ~QUEUECHANGED	
	~QUEUECHANGED,Study,12

Miscellaneous Summary

Refresh Share Index Action

This action will recreate the internal index of the media library. If for example additional media is added to a share. In order to see the new media quickly, issue this command.

```
Operation
|
#REFRESHSHAREINDEX,<player>
 |
 Command
```

Example Refresh Share Index Command

Operation	Command String
Execute #REFRESHSHAREINDEX	
	#REFRESHSHAREINDEX,Study

Ping Summary

The ping action and subsequent response are merely to verify connectivity.

Ping Action Format

```
Operation
|
#PING
 |
 Command
```

Example Ping Commands

Operation	Command String
Execute #PING	
Ping to verify connectivity	#PING
Response ~PING	
Connection is established	~PING

Error Summary

The LS1 API will respond with an error for a number of reasons detailed below.

Error Response Format

```
Operation
|
~ERROR,Error Number
 |      |
 Command Refer to "ERROR Command Specific fields" table
```

Error Command Specific fields

Error Numbers:

Description	Error Number
Unsupported Command	1
Internal Error	2
Unsupported Encoding	3