Roku

Functional Specification
Roku Control Protocol

Date:	09-Aug-07
Document Version:	2.4
Software versions:	SoundBridge 3.0.44
	WMM 3.0.44

Table of Contents

Overview

What is RCP?

This document describes the Roku Control Protocol (RCP). RCP is a control protocol implemented by the Roku SoundBridge line of digital audio players with software version 2.3 or later, and the Roku Wi-Fi Media Module (WMM), a drop-in hardware solution for implementing digital audio functionality targeted for OEMs. Remote applications can use RCP to access the digital-media functionality of those device to automate repetitive tasks, initiate and control media playback, or extend the user interface to other network-connected devices. OEMs can use RCP to control a WMM embedded in their own product and incorporate it into their existing user interface.

An Example:

RCP uses simple text-based commands to communicate client queries and commands. The following short sequence shows how an RCP client with a connected music server can list the songs from the album, "Presence," and then initiate playback of those songs starting with the first one. Note that commands sent by the client are italicized, and RCP results are in normal text.

```
SetBrowseFilterAlbum Presence
SetBrowseFilterAlbum: OK
ListSongs
ListSongs: TransactionInitiated
ListSongs: ListResultSize 7
ListSongs: Achilles Last Stand
ListSongs: For Your Life
ListSongs: Royal Orleans
ListSongs: Nobody's Fault But Mine
ListSongs: Candy Store Rock
ListSongs: Hots On For Nowhere
ListSongs: Tea For One
ListSongs: ListResultEnd
ListSongs: TransactionComplete
QueueAndPlay 0
QueueAndPlay: OK
```

Wi-Fi Media Module

The Roku Wi-Fi Media Module (WMM) allows consumer electronics device manufacturers to quickly and easily integrate networked media playback functionality into their products. WMM is based on the award-winning Roku

SoundBridge Network Media Player platform, and includes full support of its user-friendly media selection and playback functionality.

WMM (like SoundBridge) automatically discovers and communicates with media servers (like Microsoft's Windows Media Connect) running on a user's home network with no other configuration, allowing immediate access to a customer's media library with no additional software required. Internet radio stations are also fully supported, with on-board radio presets that can be stored and recalled by the user at the touch of a button. WMM implements a complete UPnP-AV media renderer, allowing third-party devices to control it using the open UPnP protocol, and it also exposes an on-board web page allowing full control from any web browser on the local network.

In short, WMM offers a ready-to-use solution for supporting network media playback in your custom consumer electronics application. The Roku Control Protocol allows a microcontroller or microprocessor in the target device to interactively access any of the WMM's built-in functionality and to retrieve the results of those actions in a synchronous or asynchronous manner. The WMM can additionally render a user interface suitable for sending to a bitmapped or character-based display, or the client can use the content enumeration and selection primitives in the WMM Control Protocol to create their own custom UI.

RCP Sessions

The RCP command protocol does not define a required physical transport for communication between an RCP-capable device and the client controller. Any communication medium that can transmit full-duplex byte stream data could be utilized. In practice, Roku devices expose RCP via the following types of connections:

1. **Serial (RS232 or I2S)** – The Wi-Fi Media Module (WMM) exposes RS232 and I2S ports via its physical connectors, and host devices can access either RCP or the SoundBridge shell console over this direct physical connection.
2. **Telnet (TCP port 5555)** – SoundBridge and WMM devices listen on TCP port 5555 at their configured IP address for incoming connections, and expose the RCP shell directly on this connection. Once connected, the device will answer with the RCP initiation sequence, "roku: ready", indicating that the connection is ready for commands.
3. **Telnet (TCP port 4444)** – SoundBridge and WMM devices listen on TCP port 4444 at their configured IP address for incoming connections, exposing the SoundBridge shell on this connection. The SoundBridge shell is a command-line interface for executing simple commands, and presents this prompt string when ready for input: "SoundBridge> ". Clients can initiate RCP sessions over this connection by executing the "mmc" shell command.

NOTE: Technically speaking, the "telnet" connections listed above do not implement the telnet protocol, but are actually just basic TCP/IP connections. Any telnet client application can connect successfully to these connections at the specified TCP ports, however, and operate without difficulty.

Terminology

When this document refers to the **RCP host**, it refers to the device that exposes the RCP interface and processes RCP commands. The RCP host can be any Roku SoundBridge device running software version 2.3 or later, or it can be a Roku Wi-Fi Media Module (WMM).

The **RCP client** is the client application or device that connects to the RCP host and sends RCP commands to the host for execution. The RCP client can be an application running on a PC or other device, a human user connected via a telnet session running commands interactively, or a microcontroller in an OEM device communicating with an embedded WMM.

An **RCP session** is a connection between an RCP host and an RCP client that lasts as long as the communications medium allows for a distinct connection. For example, a telnet (TCP) connection to a SoundBridge or WMM on port 5555 is a distinct RCP session that lasts as long as the telnet (or TCP) connection lasts. As soon as this network connection is terminated, the RCP session ends. A host microcontroller with a direct serial connection to a WMM also has one RCP session that lasts as long as the power to the WMM is connected (although it will end and restart if the WMM reboots). It is possible for a single RCP host to have several active RCP sessions at the same time.

Protocol Summary

RCP was designed with simplicity and completeness as primary requirements. Commands and results are exchanged as short transmissions across a high-speed interface like a serial port, telnet connection, or parallel interface. Each command is composed of a short ASCII command id string, generally just zero or one parameters, and the two-byte terminator CRLF. All command results from the RCP host are composed of the command-id of the client command that caused this result followed by a result string and the two-byte CRLF terminator.

(An exception to this rule exists for subscription commands, when the command result is composed of an identifier that doesn't exactly match the subscription command id. See the entries for the various subscription commands for details.)

RCP commands can be loosely categorized by the way in which they execute: *synchronous* commands, *transacted* commands, and *subscription* commands. Synchronous commands return their results immediately, and do not block the host device during execution. Transacted commands are commands that might require a long extent of time to complete, some as long as ten seconds or more! These commands generally depend on sending and receiving data from the network, which is why their completion time is non-deterministic. Transacted commands run asynchronously, or "in the background," in an RCP session, and allow the client to issue other commands or cancel the command while it is in process. Subscription commands are also asynchronous in nature, but rather than performing one atomic action, return state changes to the RCP client until the subscription is canceled.

RCP commands can also be categorized by the type of results they generate. A class of RCP commands generates a set of results returned in a list format, for example a list of music servers, or a list of music artists, or a list of songs. List results are discussed as a separate topic below. Note that the set of commands generating list results is not dependent on their categorization as synchronous, transacted, or subscription commands.

Synchronous Commands

The most basic type of synchronous command can be illustrated by the following example of the command GetTransportState, which is a command that takes no parameters and returns a single status result:

```
GetTransportState
GetTransportState: Stop
```

The RCP client sends the command id ("GetTransportState") followed by a line terminator sequence of CRLF (i.e., "\r\n" or "0d 0a" in hex), and the RCP host

responds with the result string composed of the originating command id, a colon and space separator, the status result (“Stop”), and terminating newline sequence. The results of synchronous commands are returned immediately (typically within 1 ms) by the RCP host before processing any further client commands (or returning any other RCP results).

Transacted Commands

There is another class of commands called transacted commands that can take some time to complete execution, and therefore use a slightly different execution convention. For example, one command might query a media server for a list of songs, which will require a network transaction with the server device that can take as long as several seconds to either complete or time out.

Transacted commands can return results asynchronously over the command interface throughout the lifetime of the command, so the RCP client must be designed to parse the results of these commands for some time after they are issued. The RCP client can also cancel a pending transacted command at any time during its lifetime if some user action (or other event) requires it.

In the following example, a client uses the transacted command ListArtists to get the list of media artists from a connected media server. Note the elapsed time column to the left of each transmission, which suggests a possible timing scenario for this command (although the actual timing of this command will be different and unpredictable in practice).

Elapsed Time (ms)	Transmissions
0	***ListArtists***
1	ListArtists: TransactionInitiated
400	ListArtists: ListResultSize 3
400	ListArtists: Counting Crows
400	ListArtists: Dire Straits
401	ListArtists: Led Zeppelin
401	ListArtists: ListResultEnd
401	ListArtists: TransactionComplete

NOTE: The RCP client can issue multiple transacted commands at once, however, it cannot execute multiple instances of the same command at the same time. In other words, an RCP client could issue the GetSongInfo transacted command, and while this command is still executing, it could also issue the ListArtists transacted command, and both could be pending simultaneously. However, an RCP client could not have two instances of the GetSongInfo command executing at the same time. (Try it, you’ll get the following error: “ErrorTransactionPending”.) Also, since most transacted commands return list

results, the usefulness of running them simultaneously is limited at best. (See the detailed description of list results below for more details.)

Transaction Initiation

Parsing the results of a transacted command requires some attention to detail. Transacted commands, when successfully issued, will enter the transacted state by issuing the "TransactionInitiated" result string. Once this occurs, the transaction continues processing in the background, and the RCP client can issue additional RCP commands while waiting for the results.

It is important to note that transacted commands do not always enter the transacted state. Sometimes, there are errors that can be sensed at the moment the transacted command is executed, in which case an error is returned immediately and "TransactionInitiated" is never sent! For example, the ListArtists command requires that the RCP session have an active music server connection. If there is no active server connection, ListArtists immediately returns the error result, "ErrorDisconnected", and sends nothing further.

Transaction Cancellation

All transacted commands can be canceled once the transaction has been initiated by using the CancelTransaction command. The parameter to the CancelTransaction command is the command name of the transacted command. I.e., if the RCP client in the example above decided that at 200 elapsed milliseconds it did not want to wait for the results of the ListArtists command, it could cancel it by issuing "CancelTransaction ListArtists".

Note that the success of the CancelTransaction command is non-deterministic and depends on the nature of the communications medium. For example, the RCP client could send the CancelTransaction command down a TCP connection, but the transmission of this data across the network could take a fraction of a second to complete, during which time the actual transaction could complete and send its results back! The RCP client must handle this scenario robustly. However, once a transaction has been successfully canceled (indicated by an OK result returned from the CancelTransaction command), the RCP host guarantees that it will not send any additional results from that transaction.

Transaction Progress

Some transacted commands can optionally report additional progress status while the transaction is pending. (Currently, the commands that return song lists or browse lists from music servers support this option.) This feature was added in software version 2.6.19, although to ease backward-compatibility issues with existing RCP applications, this option defaults to being inactive. Progress reporting can be activated dynamically on a per-RCP-session-basis using the SetProgressMode command.

When the RCP session is in “verbose” progress mode, the compatible transacted commands will optionally report one or more of the following states if the command takes more than about 1 second to complete:

- StatusSendingRequest
- StatusAwaitingReply
- StatusReceivingData

It is important to note that these status responses will not necessarily be sent during every transaction; they are completely optional and will only be sent if the RCP host determines that the duration of the transaction has reached a level where an end-user would benefit from a progress update. Also note that each of these responses may be sent more than once during a single command execution.

Special care must be taken when handling the StatusReceivingData response. This response can take any of the following three forms:

```
StatusReceivingData
StatusReceivingData: 120
StatusReceivingData: 120/400
```

In the first case, the RCP host has started receiving data from the server, but it has not yet received a complete result item or determined the total size of the result list. In the second case, the RCP host has received 120 items of the result list, but it cannot determine the total size of the result list. In the third case, the RCP host has received 120 items out of a total of 400 items.

Also note that when communicating with some servers (UPnP servers specifically), that the RCP host will only be able to determine the final result list size after receiving some finite number of results, so the RCP client may well see form 2 *and* form 3 of the StatusReceivingData result during the same transaction! The following example exhibits this behavior:

```
SetListResultType partial
SetListResultType: OK
SetProgressMode verbose
SetProgressMode: OK
ListContainerContents
ListContainerContents: TransactionInitiated
ListContainerContents: StatusAwaitingReply
ListContainerContents: StatusAwaitingReply
ListContainerContents: StatusReceivingData: 120
ListContainerContents: StatusReceivingData: 400/1726
ListContainerContents: StatusReceivingData: 400/1726
ListContainerContents: StatusReceivingData: 610/1726
ListContainerContents: StatusReceivingData: 880/1726
ListContainerContents: StatusReceivingData: 1150/1726
```

```
ListContainerContents: StatusReceivingData: 1420/1726
ListContainerContents: StatusReceivingData: 1690/1726
ListContainerContents: ListResultSize 1726
ListContainerContents: TransactionComplete
```

Subscription Commands

In some cases an RCP client may want status updates on state changes in the RCP host for a long period of time, spanning the interaction of many synchronous and transacted commands. For example, a client may wish to have the WMM notify it automatically every time there is a change in the transport state, e.g. when the currently playing track changes, or when there is a buffer underrun during playback. (On the other hand, some clients may prefer to poll for these status changes by requesting the current transport state with a synchronous command issued every 500 ms, e.g.)

To accommodate this kind of long-term status notification, there is a set of commands called subscription commands that causes the RCP host to asynchronously publish status updates over the control interface whenever a related change in state occurs. (Note that these status updates can interleave with results from transacted commands and in-between issuances of synchronous commands.) Subscriptions last the lifetime of the RCP session or until the RCP client explicitly cancels the subscription.

In the following example, the client subscribes to TransportState update events, and then issues the Next command (to skip to the next track) during playback.

Elapsed Time (ms)	Transmission
0	*SubscribeTransportUpdateEvents*
1	SubscribeTransportUpdateEvents: OK
1	TransportEvent: Stop
1000	*Next*
1001	Next: OK
1010	TransportEvent: TrackChange
1020	TransportEvent: Buffering
2000	TransportEvent: Play

NOTE: Unfortunately, the SubscribeTransportUpdateEvents command is un-implemented in the current version of RCP. It will be implemented in a future software update to the SoundBridge and WMM, however.

List Results

Several commands (like the aforementioned ListArtists) return a list of results. Different commands return lists of different types of objects, but each instance of a list of results is composed of the same type of object. For example, ListServers returns a list of music servers, ListAlbums returns a list of album titles, and ListSongs returns a list of songs.

Each RCP session keeps a persistent copy of the last set of list results returned by any command. This persistent set of list results can be used in future RCP commands. For example, after getting a list of music servers from the ListServers command, the RCP client can connect to one of these servers with the ServerConnect command by indicating the zero-based list position of the desired server in the list results.

It is important to note that there is only one set of list results per RCP session. So as soon as the next RCP command returning list results completes, the previous set of list results is deleted and cannot be referenced again.

Partial-Results Mode

Sometimes, these lists can be unwieldy in size, approaching 10,000 items in the most extreme cases (listing all songs in a large media library). If the entire set of 10,000 song titles was sent over a serial connection running at 115kbps and the average result string was 20 characters long, it would take 17 seconds of constant transmission to return the entire list of results! RCP includes a method for retrieving partial list results from these commands to alleviate this problem.

The client can toggle the current list result type (on a per-RCP session basis) between *full* and *partial* by issuing the SetListResultType command, which causes all subsequent commands returning list results to return results in the specified manner. When in partial result mode, commands returning list results do not return the results on completion, but just return the number of items in the list, and the client uses the command GetListResult to browse a subset of these results, specified by zero-based numeric indices. In the following example, the RCP client uses partial results to browse all songs on a media server. Note that the parameters to the GetListResult command are the starting and ending (zero-based) indices of the results in the list to display.

Elapsed Time (ms)	Transmissions
0	*SetListResultType partial*
1	SetListResultType: OK
2	*ListSongs*
2	ListSongs: TransactionInitiated
5000	ListSongs: ListResultSize 5123
5000	ListSongs: TransactionComplete
5001	*GetListResult 0 2*

5002	GetListResult: ListResultSize 3
5002	GetListResult: Ace Of Spades
5002	GetListResult: Alison
5002	GetListResult: All Mixed Up
5003	GetListResult: ListResultEnd

User Interface Models

Many RCP clients want to allow interactive access to all the digital-media features of the RCP host device. That is, they want their users to be able to interactively start and stop music playback, select songs by different categories, even connect to different music servers. To implement this, the client device needs to present a user interface to their users that can display lists of options for the user to choose from, and allow random-access to transport commands like “play” and “pause”.

RCP offers two different models with which an RCP client can construct a user interface: client-generated UI, and RCP host-generated UI.

Client-Generated UI

In the client-generated UI model, the RCP client creates its own custom user interface using whatever type of display is available to it, be that a television display, computer monitor, bitmapped or character-based LCD display, or other! The RCP client issues RCP commands to list the options that it wants to present to the user, and then issues further RCP commands to act on the user’s selections. For example, to choose a song for playback, the RCP client would issue the ListSongs command, then format the results into a list of songs on the device’s display and let the user pick one, then it could send the QueueAndPlay command to select one of those songs for playback.

RCP Host-Generated UI

In the host-generated UI model, the RCP client can take advantage of the existing award-winning user interface of the SoundBridge network media player, which is built-in to all RCP host devices, including WMM. The RCP host can create and maintain the user interface, and the RCP client can just echo the current contents of the “display data” on its own physical display, and route user key presses back to the RCP host for it to take actions on.

In this model, the RCP host device creates the user interface and exports it through the GetDisplayData command, which returns the contents of the current display in a bitmapped or character-based format, depending on the configuration of the RCP host device. To interact with the user, the client device issues the command IrDispatchCommand to notify the RCP host when the user hits buttons like the arrow-left, arrow-right, and PLAY buttons.

To keep the physical display up to date with changes to the RCP host-generated UI, you can subscribe to display-update events (to receive notification whenever

the UI has changed), you can poll for a display-update counter, or you can simply download the display data (using the command GetDisplayData) several times a second.

RCP Limitations and Future Directions

RCP is an evolving protocol, as is the software running on SoundBridge and WMM devices. Roku is constantly adding features and expanding the capabilities of its devices, and offers end-customers the ability to easily upgrade the software over the internet. As new features are added to the software, new commands and constructs are added to RCP to allow access to these features. Additionally, Roku works closely with its OEM and retail customers to improve RCP to meet their needs.

As such, there are known areas where the functionality of RCP is planned for improvement in future software releases. Until then, we would like to mention some of these known areas and discuss our future plans for them and possible work-arounds that may be applied at the moment.

Getting Information About the Connected Server

Historically, the ability to get information about the connected server has been limited. However, starting with version 3.0.23 of the SoundBridge software, the GetActiveServerInfo and ServerGetCapabilities commands have been added. These two commands allow the RCP client to find the name, protocol, and search/browse capabilities of the connected server. Please see those sections for details.

Standby Mode

Some RCP clients would like to know if the RCP host is in standby mode. Standby mode is accessed when the RCP host-generated UI processes a power button IR command. In this mode, the RCP host disconnects from any currently connected music server and displays the current time on its display.

Prior to software version 3.0.23, it was not possible to query the unit's standby mode status, or "power state". However, with 3.0.23, the commands GetPowerState and SetPowerState have been provided to allow clients to query the host's power state, as well as to set the host's power state.

NOTE: There is a bug in software 2.5.171 where if the RCP host is in standby mode and an RCP client executes the ServerLaunchUI command, then the unit will connect to the server but it will get into a state where it cannot enter standby mode again during that boot session.

Other Areas For Improvement

RCP has only a limited ability to set the user presets. See the section "Commands for Using Presets" for more information. Roku plans on adding

commands to let the RCP client set all types of user presets, not just internet radio presets.

RCP support for socket commands only supports simple TCP sockets at present. See the section “Socket Commands for Generalized Network Communication”. Roku plans on adding full socket support including datagram (UDP) sockets and listening stream (TCP) sockets.

RCP is planned to offer support for drawing on the display using drawing primitive commands. This capability is currently available from the SoundBridge shell command sketch, but we would like to roll this into RCP eventually.

When connecting to servers that support a container hierarchy, it is possible through the RCP-host user interface to queue for playback all the songs below a certain container. E.g., the user can browse to an “Artists” container, highlight the name of an artist, and then hit the PLAY button to play all the songs within that container, even if it has album sub-containers within it. The only way to duplicate this process in RCP is to manually recurse through all the sub-containers and add all songs with the NowPlayingInsert command. This is unnecessarily complex. We would like to add a command to handle this procedure automatically.

There is also a need for an error-detection protocol for guaranteeing the delivery of RCP transactions on a non-error-correcting connection, like RS232. Telnet/TCP connections have error-detection built-in to the communication protocol, so clients can assume the connection is dependable.

SampleCommand

Syntax:

The syntax of the command goes here. 'n' stands for a numeric parameter, often a zero-based index into previous list results. Optional parameters are listed in [square brackets], and parameters where one token can be chosen from a finite list are|vertical-bar|delimited.

Classification:

Synchronous, Transacted, or Subscription.

Result Type:

If the function returns results that can be used by subsequent commands during the RCP session, like list results, they are listed here. Also, if the function returns an object with many fields, it is described here. Functions that just return status tokens are considered to have no result type.

Responses:

All possible response tokens are listed here, with a short description of each. Note that future software releases might add additional failure responses, so assume unknown responses indicate failure.

Example:

A single invocation of the command is listed here with an example result. For example:

SetServerFilter daap upnp
SetServerFilter: OK

Finally, additional description of the command goes here.

Protocol Control Commands

These commands allow the RCP client to control the operation of the current RCP session.

- CancelTransaction
- GetListResultType
- SetListResultType
- GetListResult
- DeleteList
- GetDataResultType
- SetDataResultType
- GetProgressMode
- SetProgressMode

These commands are described in the pages that follow.

CancelTransaction

Syntax:

CancelTransaction command-id

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid command id
ErrorTransactionNotPending – not pending
OK – indicates success

Example:

ListSongs
ListSongs: TransactionInitiated
CancelTransaction ListSongs
ListSongs: TransactionCanceled
CancelTransaction: OK

CancelTransaction is used to abort the asynchronous transaction listed in the command id parameter. Any transacted command can be canceled using CancelTransaction.

After using CancelTransaction to abort an asynchronous command, that command's results are in an undefined state. So, using commands that depend on the aborted command's results will have undefined behavior. However, subsequent commands that don't depend on that command's results will behave normally.

NOTE: The success of canceling a transacted command is non-deterministic, even if that command has not sent a TransactionComplete result at the moment the RCP client issues a CancelTransaction command. See the "Transacted Commands" section above.

GetListResultType

Syntax:

GetListResultType

Classification:

Synchronous

Result Type:

Returns a list result type token.

Responses:

No responses

Example:

GetListResultType
GetListResultType: full

GetListResultType returns the current list result type setting for the RCP session. Use SetListResultType to modify this value. In full results mode, commands that generate list results return all results at command completion. In partial results mode, these commands only return a ListResultSize value, the number of results available for accessing using GetListResult.

See also the section on “List Results” above.

HISTORICAL NOTE: GetListResultType was added in software version 2.6.19. TODO: Note when merged into 2.7.

SetListResultType

Syntax:

SetListResultType full|partial

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid result type
OK – indicates success

Example:

SetListResultType partial
SetListResultType: OK
ListSongs
ListSongs: TransactionInitiated
ListSongs: ListResultSize 1719
ListSongs: TransactionComplete

SetListResultType allows you to switch between full results mode and partial results mode. In full results mode, commands that generate list results return all results at command completion. In partial results mode, these commands only return a ListResultSize value, the number of results available for accessing using GetListResult.

See also the section on “List Results” above.

GetListResult

Syntax:

GetListResult start-index end-index

Classification:

Synchronous

Result Type:

Returns a list of objects (type depends on the last command returning list results that completed successfully)

Responses:

ParameterError – invalid indices or no list reslts available

Example:

ListSongs
ListSongs: TransactionInitiated
ListSongs: ListResultSize 1719
ListSongs: TransactionComplete
GetListResult 10 12
GetListResult: ListResultSize 3
GetListResult: Magic Touch
GetListResult: Rag Doll
GetListResult: Hangman Jury
GetListResult: ListResultEnd

GetListResult will list the list results of the last command executed in the current RCP session that returned list results. Note that GetListResult may be used both in "partial results mode" and "full results mode."

See the section on "List Results" above for more information.

DeleteList

Syntax:

DeleteList

Classification:

Synchronous

Result Type:

No results

Responses:

ErrorNoListResults
OK – indicates success

Example:

DeleteList
DeleteList: OK

DeleteList will delete the current list results of the RCP session. Use of this command is generally unnecessary, as previous list results are always discarded each time a new command that generates list results is executed. However, some commands (see SetWiFiNetworkSelection) may require use of this command to remove ambiguity on a numeric parameter.

GetDataResultType

Syntax:

GetDataResultType

Classification:

Synchronous

Result Type:

Returns a data result type token

Responses:

No responses

Example:

GetDataResultType
GetDataResultType: hex

GetDataResultType returns the current data result type for the RCP session. Use the command SetDataResultType to change this value. In binary mode, data results are streamed to the RCP client as an un-modified binary byte stream. In hex mode, data results are streamed to the RCP client as a stream of hex digits.

HISTORICAL NOTE: GetDataResultType was added in software version 2.6.19. TODO: Note when merged into 2.7.

SetDataResultType

Syntax:

SetDataResultType binary|hex

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid result type
OK – indicates success

Example:

SetDataResultType binary
SetDataResultType: OK

SetDataResultType allows the RCP client to change the way that commands return data results. In binary mode, data results are streamed to the RCP client as an un-modified binary byte stream. In hex mode, data results are streamed to the RCP client as a stream of hex digits.

GetProgressMode

Syntax:

GetProgressMode

Classification:

Synchronous

Result Type:

Returns a progress mode token

Responses:

No responses

Example:

GetProgressMode
GetProgressMode: off

GetProgressMode returns the current progress mode of the RCP session. The progress mode is either “verbose” or “off”. In verbose mode, many transacted commands will output additional progress reports before reaching completion. Use the command SetProgressMode to modify this value.

HISTORICAL NOTE: GetProgressMode was added in software version 2.6.19. TODO: Note when merged into 2.7.

SetProgressMode

Syntax:

SetProgressMode verbose|off

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid progress mode
OK – indicates success

Example:

SetProgressMode verbose
SetProgressMode: OK

SetProgressMode changes the progress mode of the current RCP session. In verbose mode, several transacted commands return additional progress responses during their execution. This command exists to allow RCP client applications written before these progress responses were added to continue to work without modification. All RCP client applications should be written to accept these progress responses for future compatibility.

The following commands output additional progress responses when the progress mode is set to verbose:

- ListSongs
- ListAlbums
- ListArtists
- ListComposers
- ListGenres
- ListLocations
- ListMediaLanguages
- ListPlaylists
- ListPlaylistSongs
- ListContainerContents
- SearchSongs
- SearchArtists
- SearchAlbums
- SearchComposers
- SearchAll

HISTORICAL NOTE: SetProgressMode was added in software version 2.6.19. TODO: Note when merged into 2.7.

Host Configuration Commands

The commands in this section may be used to administer the basic setup of the RCP host, including language settings, network settings, time and date setup, detecting Safe-Mode, and performing software upgrades.

Initial Setup

Many of these commands deal with the Initial Setup process. This process is required on units that have been reset to factory defaults or have just been run for the first time out of the box. During the initial setup procedure, the unit must be configured with a language, time zone, and region settings. Most basic commands will not operate before initial setup is complete!

NOTE: Not all types of SoundBridge units or WMMs require the same initial setup steps. Use the GetRequiredSetupSteps command to find out which steps are required on the RCP host you are using.

After completing the settings required by Initial Setup, the RCP client can use the SetInitialSetupComplete command to finish the Initial Setup process and make the RCP host ready to use.

The following host configuration commands are available:

- GetInitialSetupComplete
- SetInitialSetupComplete
- GetRequiredSetupSteps
- ListLanguages
- GetLanguage
- SetLanguage
- ListRegions
- SetRegion
- GetTermsOfServiceUrl
- AcceptTermsOfService
- GetIfConfig
- GetLinkStatus
- GetIPAddress
- GetMACAddress
- ListWiFiNetworks
- GetWiFiNetworkSelection
- SetWiFiNetworkSelection
- SetWiFiPassword
- WiFiNetworkConnect
- GetConnectedWiFiNetwork
- GetWiFiSignalQuality
- GetTime
- GetDate

- SetTime
- SetDate
- ListTimeZones
- GetTimeZone
- SetTimeZone
- GetSoftwareVersion
- GetBootMode
- Reboot
- CheckSoftwareUpgrade
- ExecuteSoftwareUpgrade
- ResetToFactoryDefaults
- GetFriendlyName
- SetFriendlyName

GetInitialSetupComplete

Syntax:

GetInitialSetupComplete

Classification:

Synchronous

Result Type:

No results

Responses:

Complete
NotComplete

Example:

GetInitialSetupComplete
GetInitialSetupComplete: Complete

GetInitialSetupComplete returns Complete or NotComplete to inform the RCP client if Initial Setup has been completed on this RCP host.

SetInitialSetupComplete

Syntax:

SetInitialSetupComplete

Classification:

Synchronous

Result Type:

No results

Responses:

ErrorAlreadySet – Initial Setup was already complete
NotComplete – setup steps are not complete
OK – indicates success

Example:

SetInitialSetupComplete
SetInitialSetupComplete: OK

Use SetInitialSetupComplete to complete the Initial Setup process and make the RCP host ready to use. Before executing this command, all the required setup choices must have been made.

GetRequiredSetupSteps

Syntax:

GetRequiredSetupSteps

Classification:

Synchronous

Result Type:

No results

Responses:

Language
Region
TimeZone

Example:

GetRequiredSetupSteps
GetRequiredSetupSteps: Language
GetRequiredSetupSteps: TimeZone
GetRequiredSetupSteps: Region
GetRequiredSetupSteps: TermsOfService
GetRequiredSetupSteps: OK

GetRequiredSetupSteps returns a response token representing each of the required steps for completing Initial Setup on the RCP host. If the Language token is returned, then the RCP client should use the ListLanguages and SetLanguage commands to select a language. If the TimeZone token is returned, the RCP client should use the ListTimeZones and SetTimeZone commands to select a time zone. If the Region token is returned, the RCP client should use the ListRegions and SetRegion commands to select a region. If the TermsOfService token is returned, the RCP client is required use GetTermsOfServiceUrl and AcceptTermsOfService to accept the terms of service.

ListLanguages

Syntax:

ListLanguages

Classification:

Synchronous

Result Type:

Returns a list of language names

Responses:

No responses

Example:

```
ListLanguages
ListLanguages: ListResultSize 5
ListLanguages: Svenska
ListLanguages: Français
ListLanguages: Espanol
ListLanguages: Deutsch
ListLanguages: English
ListLanguages: ListResultEnd
```

ListLanguages lists all languages supported for use in the UI display.

GetLanguage

Syntax:

GetLanguage

Classification:

Synchronous

Result Type:

Returns a language name

Responses:

No responses

Example:

GetLanguage
GetLanguage: English

GetLanguage displays the name of the language which is used for the UI display text.

SetLanguage

Syntax:

SetLanguage n|language-name

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid language specified
GenericError – failed to set language
OK – indicates success

Example:

SetLanguage 0
SetLanguage: OK

SetLanguage sets the language which is to be used for the UI display text to the language of index n. The index value is with respect to the list output by the ListLanguages command. The parameter can also be a language name as returned by ListLanguages.

ListRegions

Syntax:

ListRegions

Classification:

Synchronous

Result Type:

Returns a list of region names

Responses:

No responses

Example:

```
ListRegions
ListRegions: ListResultSize 5
ListRegions: USA
ListRegions: Canada
ListRegions: France
ListRegions: Japan
ListRegions: United Kingdom
ListRegions: ListResultEnd
```

ListRegions returns a list of region names for use with the SetRegion command.

SetRegion

Syntax:

SetRegion n

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid index or list results
ErrorUnsupported – region setting not supported on this host
ErrorAlreadySet – region can only be set once
GenericError – unexpected error
OK – indicates success

Example:

SetRegion 1
SetRegion: OK

SetRegion sets the region code on RCP hosts that require a region setting. (Use GetRequiredSetupSteps to find out if region setting is required.) The index parameter n refers to the zero-based index of a region in the list of regions returned by ListRegions, which must have been the last executed command returning list results by the current RCP session.

NOTE: There is no way to query the region value on a unit that has already had the region set.

GetTermsOfServiceUrl

Syntax:

GetTermsOfServiceUrl

Classification:

Synchronous

Result Type:

No results

Responses:

Url for terms of service.

Example:

```
GetTermsOfServiceUrl
GetTermsOfServiceUrl: http://rokulabs.com/tos
```

The URL returned is where the user can be pointed to access the terms of service.

Typically, the client device would display an on-screen message asking the user to visit the specified URL, read the Terms of Service, and then click to indicate approval, at which point the client would execute AcceptTermsOfService.

AcceptTermsOfService

Syntax:

AcceptTermsOfService

Classification:

Synchronous

Result Type:

No results

Responses:

Accepted terms of service from ***URL***

Example:

AcceptTermsOfService
AcceptTermsOfService: Accepted terms of service from http://rokulabs.com/tos

GetIfConfig

Syntax:

GetIfConfig

Classification:

Synchronous

Result Type:

Returns lines of text representing the output of the ifconfig shell command

Responses:

No responses

Example:

```
GetIfConfig
GetIfConfig: B200   Interface: 0 (kWireless) has
a dhcp assigned address
GetIfConfig:  MAC Address: 00:0D:4B:05:02:8F
GetIfConfig:   IP Address: 172.17.2.234
GetIfConfig:      Netmask: 255.255.0.0
GetIfConfig:      Gateway: 172.17.0.2
GetIfConfig:        DNS 1: 172.17.0.2
GetIfConfig:        DNS 2: 172.17.0.12
```

GetIfConfig returns the results of the ifconfig shell command.

GetLinkStatus

Syntax:

GetLinkStatus [enet|wlan|loop|n]

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError
ErrorNotFound
Link
NoLink

Example:

GetLinkStatus enet
GetLinkStatus: Link

GetLinkStatus returns Link or NoLink to inform the RCP client if there is an active network link on the interface identified by the command parameter. Network interfaces may be identified by type (enet, wlan, or loop for Ethernet, Wireless, or Loopback, respectively) or by number. RCP hosts currently have one or two network interfaces numbered 0 and 1. If no parameter is given, the command examines the first interface found (useful to determine that the client has network connectivity).

GetIPAddress

Syntax:

GetIPAddress [enet|wlan|loop|n]

Classification:

Synchronous

Result Type:

IPV4 address in dotted decimal notation

Responses:

ParameterError
ErrorNotFound

Example:

GetIPAddress 0
GetIPAddress: 192.168.0.101

GetIPAddress returns the IP address on the network interface identified by the command parameter. Network interfaces may be identified by type (enet, wlan, or loop for Ethernet, Wireless, or Loopback, respectively) or by number. RCP hosts currently have one or two network interfaces numbered 0 and 1. If no parameter is given, the command examines the first interface found.

GetMACAddress

Syntax:

GetMACAddress [enet|wlan|loop|upgr|n]

Classification:

Synchronous

Result Type:

MAC address string

Responses:

ParameterError
ErrorNotFound

Example:

GetMACAddress
GetMACAddress: 00:0D:4B:01:02:03

GetMACAddress returns the hardware MAC address on the network interface identified by the command parameter. Network interfaces may be identified by type or by number.

The enet, wlan, and loop types represent Ethernet, Wireless, or Loopback, respectively. The upgr type returns the MAC address that is used to identify a unit to the upgrade server. For most SoundBridge models, this is the Ethernet MAC address, but for models with no Ethernet, it is the Wi-Fi MAC.

The MAC address for a given interface can be queried with these types even if that interface is currently inactive (*e.g.* even with Ethernet disconnected and the interface thus disabled, ***GetMACAdddress enet*** will still return the Ethernet MAC). If the interface is simply absent, however, as on models without Ethernet hardware, then the **ErrorNotFound** response will be returned.

RCP hosts currently have one or two network interfaces numbered 0 and 1, so these are the values allowed for the numeric argument. As with the types above, the numeric arguments will return the information even if the selected interface is inactive.

If no parameter is given, the command applies to the first *active* interface found. A disabled interface's MAC (*e.g.* for an unplugged Ethernet port) will not be returned by this command when no argument is specified. Use the explicit arguments above.

ListWiFiNetworks

Syntax:

ListWiFiNetworks

Classification:

Transacted

Result Type:

Returns a list of network names

Responses:

ErrorInitialSetupRequired – initial setup not complete
ErrorNoWiFiInterfaceFound – no wi-fi hardware found
ErrorWiFiInterfaceDisabled

Example:

ListWiFiNetworks
ListWiFiNetworks: TransactionInitiated
ListWiFiNetworks: ListResultSize 2
ListWiFiNetworks: testlan
ListWiFiNetworks: nirvana
ListWiFiNetworks: ListResultEnd
ListWiFiNetworks: TransactionComplete

ListWiFiNetworks returns a list of Wi-Fi network names (SSIDs) found by scanning for networks that broadcast their existence. The scanning process generally takes a few seconds to complete.

If there is an initial error condition found (see responses listed above), the error will be returned immediately and no transaction will be initiated.

GetWiFiNetworkSelection

Syntax:

GetWiFiNetworkSelection

Classification:

Synchronous

Result Type:

Returns a wireless network name

Responses:

NoSelection

Example:

GetWiFiNetworkSelection
GetWiFiNetworkSelection: my network

GetWiFiNetworkSelection returns the current wi-fi network SSID selection. Note that a Wi-Fi network may be selected even if it is not currently connected, and even if there is no Wi-Fi support on the RCP host. GetWiFiNetworkSelection returns NoSelection if there is no SSID selected for connection.

SetWiFiNetworkSelection

Syntax:

SetWiFiNetworkSelection ssid|n

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError
OK – indicates success

Example:

SetWiFiNetworkSelection 1
SetWiFiNetworkSelection: OK

SetWiFiNetworkSelection selects an SSID for the RCP host to connect to. The SSID is either passed by name directly as a string parameter, or the parameter is a numeric zero-based index into the list results of a previous call to ListWiFiNetworks, which must be the last command executed returning list results over the current RCP session.

Since a numeric index parameter could also be a valid SSID (e.g., an SSID of "1"), the RCP client can execute the DeleteList command to delete the current list results of a call to ListWiFiNetworks in order to remove this ambiguity. (If the parameter is a valid index into an existing wi-fi network list, then the SSID from the list is used.)

The RCP host will not attempt to connect to the selected Wi-Fi network when this function is executed; this function just sets the Wi-Fi network selection on the host, and in Release-Mode, it also persists the selection to flash ROM. If the network requires a password, it must be set after selecting the SSID with the command SetWiFiPassword.

In Safe-Mode, use the command WiFiNetworkConnect to connect to a Wi-Fi network once you have set the appropriate SSID and (optional) password.

SetWiFiPassword

Syntax:

SetWiFiPassword password

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no password specified
NoSelection – no SSID selected
GenericError – internal error
OK

Example:

SetWiFiPassword 0XAABBCCDDEE
SetWiFiPassword: OK

SetWiFiPassword will set a connection password for connecting to a wi-fi network. A SSID must have been previously selected, through the host-generated UI or with the SetWiFiNetworkConnection command. The password parameter is interpreted as an ASCII string unless it begins with "0X", in which case it is assumed to be a string of hex digits.

RCP host hardware configurations currently support 40-bit and 128-bit WEP. (128-bit WEP uses 104-bit keys, but is commonly called "128-bit WEP.") RCP clients may set the password using a hex string or an ASCII string of the appropriate length. Refer to the following table for examples of setting the WEP key using hex or ASCII strings for 40- and 128-bit WEP.

Configuration	Password parameter format
40-bit ASCII	abcde
40-bit hex	0XAABBCCDDEE
128-bit ASCII	abcdefghijklm
128-bit hex	0X00112233445566778899AABBCC

NOTE: Please keep the hex string formatted with upper-case hex digits.

NOTE: RCP host hardware configurations do not currently support 256-bit WEP or WPA security.

WiFiNetworkConnect

Syntax:

WiFiNetworkConnect

Classification:

Synchronous

Result Type:

No results

Responses:

ErrorInitialSetupRequired
ErrorNotFound – no Wi-Fi interface detected
ErrorCantReconfigure – already connected
NoSelection – SSID not set
OK – indicates success

Example:

WiFiNetworkConnect
WiFiNetworkConnect: OK

WiFiNetworkConnect is only available in Safe-Mode and is used to connect to a Wi-Fi network in order to perform a software upgrade. Before connecting, the initial setup process must be complete, and the network SSID and (optional) password must be set using commands SetWiFiNetworkSelection and SetWiFiPassword.

GetConnectedWiFiNetwork

Syntax:

GetConnectedWiFiNetwork

Classification:

Synchronous

Result Type:

Returns a network name

Responses:

GenericError

Example:

GetConnectedWiFiNetwork
GetConnectedWiFiNetwork: my network

GetConnectedWiFiNetwork will return the SSID network name of the currently connected Wi-Fi network. Note that not all RCP hosts support this function, in which case GenericError will be returned.

GetWiFiSignalQuality

Syntax:

GetWiFiSignalQuality

Classification:

Synchronous

Result Type:

Returns quality results

Responses:

OK

Example:

```
GetWiFiSignalQuality
GetWiFiSignalQuality: quality: 94
GetWiFiSignalQuality: signal: -46
GetWiFiSignalQuality: noise: -1
GetWiFiSignalQuality: OK
```

TBD

GetTime

Syntax:

GetTime [verbose]

Classification:

Synchronous

Result Type:

Returns a time string in the format HH:MM:SS or formatted for the current locale

Responses:

TimeNotSet

Example:

GetTime
GetTime: 09:35:33
GetTime verbose
GetTime: 9:35 AM

GetTime returns the current time in the format HH:MM:SS. If the RCP host's time is not set then the error result TimeNotSet is returned. GetTime will optionally format the time in the locale specified by the user language setting if you pass the optional "verbose" parameter.

GetDate

Syntax:

GetDate [verbose]

Classification:

Synchronous

Result Type:

Returns a date string in the format MM-DD-YYYY or formatted for the current locale

Responses:

TimeNotSet

Example:

GetDate
GetDate: 09-02-2006
GetDate verbose
GetDate: Saturday, September 2, 2006

GetDate returns the current date in the format MM-DD-YYYY. If the RCP host's time is not set then the error result TimeNotSet is returned. GetDate will optionally format the date in the locale specified by the user language setting if you pass the optional "verbose" parameter.

SetTime

Syntax:

SetTime HH:MM:SS

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError
GenericError – time subsystem error
OK – indicates success

Example:

SetTime 12:40:00
SetTime: OK

SetTime sets the current time of the RCP host. If the date has not yet been set, SetTime has the side-effect of setting the date to Jan 1, 2006.

NOTE: Each RCP host has a Network Time Protocol (NTP) subsystem that gets the current time and date from a time server at regular intervals. The current time will be reset by the NTP subsystem the next time it successfully connects to an Internet time server. There is currently no way to disable this behavior.

SetDate

Syntax:

SetDate MM-DD-YYYY

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError
GenericError – time subsystem error
OK – indicates success

Example:

SetDate
SetDate: Complete

SetDate sets the current date of the RCP host. If the time has not yet been set, SetDate has the side-effect of setting the time to 00:00:00 (midnight).

NOTE: Each RCP host has a Network Time Protocol (NTP) subsystem that gets the current time and date from a time server at regular intervals. The current date will be reset by the NTP subsystem the next time it successfully connects to an Internet time server. There is currently no way to disable this behavior.

ListTimeZones

Syntax:

ListTimeZones

Classification:

Synchronous

Result Type:

Returns a list of time zones

Responses:

No responses

Example:

```
ListTimeZones
ListTimeZones: ListResultSize 63
ListTimeZones: US Eastern Time
ListTimeZones: US Central Time
ListTimeZones: US Mountain Time
ListTimeZones: US Pacific Time
ListTimeZones: Alaska Time
ListTimeZones: Hawaii-Aleutian Time with no Daylight Savings (Hawaii)
ListTimeZones: Hawaii-Aleutian Time with Daylight Savings
ListTimeZones: US MT without Daylight Savings Time (Arizona)
ListTimeZones: US ET without Daylight Savings Time (East Indiana)
ListTimeZones: Atlantic Time
ListTimeZones: Brazil Time (Sao Paulo)
ListTimeZones: Newfoundland Time
ListTimeZones: Azores Time
ListTimeZones: London/Dublin Time
ListTimeZones: Western European Time
ListTimeZones: Central European Time
ListTimeZones: Eastern European Time
ListTimeZones: Moscow Time
ListTimeZones: Delta Time Zone (Samara)
ListTimeZones: Echo Time Zone (Yekaterinburg)
ListTimeZones: Indian Standard Time
ListTimeZones: Foxtrot Time Zone (Omsk)
ListTimeZones: Japanese Standard Time
ListTimeZones: Christmas Island Time (Australia)
ListTimeZones: Australian Western Time
ListTimeZones: Australian Central Time
ListTimeZones: Australian Central Time without Daylight Savings Time
(Darwin)
ListTimeZones: Australian Eastern Time
ListTimeZones: Australian Eastern Time without Daylight Savings Time
(Brisbane)
ListTimeZones: Norfolk (Island) Time (Australia)
ListTimeZones: New Zealand Time (Auckland)
ListTimeZones: Chatham Time Zone
ListTimeZones: Yankee Time Zone (Fiji)
ListTimeZones: X-ray Time Zone (Pago Pago)
ListTimeZones: Greenwich Mean Time
ListTimeZones: 1 hour ahead of Greenwich Mean Time
ListTimeZones: 2 hours ahead of Greenwich Mean Time
ListTimeZones: 3 hours ahead of Greenwich Mean Time
ListTimeZones: 4 hours ahead of Greenwich Mean Time
ListTimeZones: 5 hours ahead of Greenwich Mean Time
ListTimeZones: 6 hours ahead of Greenwich Mean Time
ListTimeZones: 7 hours ahead of Greenwich Mean Time
ListTimeZones: 8 hours ahead of Greenwich Mean Time
ListTimeZones: 9 hours ahead of Greenwich Mean Time
```

```
ListTimeZones: 10 hours ahead of Greenwich Mean Time
ListTimeZones: 11 hours ahead of Greenwich Mean Time
ListTimeZones: 12 hours ahead of Greenwich Mean Time
ListTimeZones: 13 hours ahead of Greenwich Mean Time
ListTimeZones: 14 hours ahead of Greenwich Mean Time
ListTimeZones: 1 hour behind Greenwich Mean Time
ListTimeZones: 2 hours behind Greenwich Mean Time
ListTimeZones: 3 hours behind Greenwich Mean Time
ListTimeZones: 4 hours behind Greenwich Mean Time
ListTimeZones: 5 hours behind Greenwich Mean Time
ListTimeZones: 6 hours behind Greenwich Mean Time
ListTimeZones: 7 hours behind Greenwich Mean Time
ListTimeZones: 8 hours behind Greenwich Mean Time
ListTimeZones: 9 hours behind Greenwich Mean Time
ListTimeZones: 10 hours behind Greenwich Mean Time
ListTimeZones: 11 hours behind Greenwich Mean Time
ListTimeZones: 12 hours behind Greenwich Mean Time
ListTimeZones: 13 hours behind Greenwich Mean Time
ListTimeZones: 14 hours behind Greenwich Mean Time
ListTimeZones: ListResultEnd
```

ListTimeZones lists the names of the time zones available on the RCP host.

GetTimeZone

Syntax:

GetTimeZone

Classification:

Synchronous

Result Type:

Returns the name of the current time zone

Responses:

GenericError – time subsystem error

Example:

GetTimeZone
GetTimeZone: US Pacific Time

GetTimeZone returns the name of the current time zone.

SetTimeZone

Syntax:

SetTimeZone n

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError
GenericError – time subsystem error
OK – indicates success

Example:

SetTimeZone 0
SetTimeZone: OK

SetTimeZone will set the current time zone on the RCP host. The required parameter is a zero-based numeric index into the current RCP session list results, which must be the results of the ListTimeZones command.

GetSoftwareVersion

Syntax:

GetSoftwareVersion

Classification:

Synchronous

Result Type:

Returns software version string

Responses:

No responses

Example:

GetSoftwareVersion
GetSoftwareVersion: 2.5.173

GetSoftwareVersion returns a string representing the software version running on the RCP host.

GetBootMode

Syntax:

GetBootMode

Classification:

Synchronous

Result Type:

No results

Responses:

Release
Safe

Example:

GetBootMode
GetBootMode: Release

GetBootMode returns the current boot mode of the RCP host.

Release Mode is the standard boot mode, and Safe Mode is the back-up boot mode that is run on boot when the release-mode software image is determined to be corrupted. This could happen e.g., if the power is removed in the middle of a software upgrade.

RCP support in Safe Mode is limited to commands to complete initial setup and upgrade the software so the unit can be rebooted into Release Mode.

HISTORICAL NOTE: Shipping retail hardware configurations do not have any support for RCP in Safe Mode. The latest development version of Safe Mode with RCP support is available to authorized customers on request.

CheckSoftwareUpgrade

Syntax:

CheckSoftwareUpgrade [local]

Classification:

Transacted

Result Type:

Returns software version of upgrade (for network check only)

Responses:

UpgradeAvailable
NoUpgradeAvailable
ErrorInitialSetupRequired
ErrorNoMACAddress
ErrorNoDNS
ErrorDNSFailed
ErrorHTTPResult
ErrorInvalidServerResponse

Example:

CheckSoftwareUpgrade
CheckSoftwareUpgrade: TransactionInitiated
CheckSoftwareUpgrade: 2.6.12
CheckSoftwareUpgrade: TransactionComplete

CheckSoftwareUpgrade will check for available software upgrades for the RCP host. By default the check is performed by contacting the upgrade servers on the Internet. By passing the optional "local" parameter, the check will instead be made on removable media inserted into the local RCP host. (Not all RCP hosts have media slots on them.)

If an upgrade is found when checking for upgrades over the network, this command will return the software version of the new software. If an upgrade is found on a local storage device, however, only the UpgradeAvailable result will be sent. When an upgrade is not available, the NoUpgradeAvailable result will be sent.

ExecuteSoftwareUpgrade

Syntax:

ExecuteSoftwareUpgrade [local]

Classification:

Transacted

Result Type:

No results

Progress Responses:

CheckingUpgradeAvailability
CopyingUpgrade
DownloadingUpgrade
VerifyingMD5
ErasingFlash
BurningFlash
VerifyingFlash
SavingAdditionalData
UpgradeComplete

Error Responses:

NoUpgradeAvailable
ErrorAwaitingPostUpgradeReboot
ErrorUpgradeInProgress
ErrorCantRunWithServerConnected
ErrorNoMACAddress
ErrorNoDNS
ErrorDNSFailed
ErrorTimeoutReceivingData
ErrorHTTPResult
ErrorOutOfMemory
ErrorUpgradeTooLarge
ErrorFileNotFound
ErrorFileReadError
ErrorMD5VerificationFailed
ErrorFlashEraseFailed
ErrorFlashBurnFailed
ErrorFlashVerifyFailed

Example:

ExecuteSoftwareUpgrade
ExecuteSoftwareUpgrade: TransactionInitiated
ExecuteSoftwareUpgrade:
CheckingUpgradeAvailability
ExecuteSoftwareUpgrade: DownloadingUpgrade: 0

```
ExecuteSoftwareUpgrade: DownloadingUpgrade: 1
ExecuteSoftwareUpgrade: DownloadingUpgrade: 2
ExecuteSoftwareUpgrade: DownloadingUpgrade: 3
[steps removed]
ExecuteSoftwareUpgrade: DownloadingUpgrade: 98
ExecuteSoftwareUpgrade: DownloadingUpgrade: 99
ExecuteSoftwareUpgrade: DownloadingUpgrade: 100
ExecuteSoftwareUpgrade: VerifyingMD5
ExecuteSoftwareUpgrade: ErasingFlash
ExecuteSoftwareUpgrade: BurningFlash
ExecuteSoftwareUpgrade: VerifyingFlash
ExecuteSoftwareUpgrade: SavingAdditionalData
ExecuteSoftwareUpgrade: UpgradeComplete
ExecuteSoftwareUpgrade: TransactionComplete
```

ExecuteSoftwareUpgrade will upgrade the software stored on the RCP host to the latest available version. The new software may be found by checking the software upgrade servers on the Internet (default) or by searching local removable media slots on the RCP host (using the optional “local” parameter).

Upgrading the software is a process that takes up to a full minute to complete, depending on the hardware configuration of the RCP host. Caution should be taken to not reboot the unit during a software upgrade. In fact, Roku recommends not executing any unnecessary commands on the RCP host while a software upgrade is in process. To this end, several commands will return ErrorUpgradeInProgress if you attempt to execute them (from this or any other RCP session) while an upgrade is in progress.

The RCP host will send progress updates on the upgrade while it executes to the RCP client. (See the listing of progress responses in the command summary above.) Not all of these progress responses will always be sent, depending on the configuration and any errors that may be encountered. E.g., CopyingUpgrade is only sent for local software upgrades, while DownloadingUpgrade is only sent for network software upgrades.

When downloading (or copying) the upgrade image file, the RCP host will send multiple progress responses indicating the percent completion of the process, indicated by a decimal number from 0 to 100. The RCP client may present this information to the end-user or safely ignore it. Progress reports have the form:

```
ExecuteSoftwareUpgrade: DownloadingUpgrade: 50
ExecuteSoftwareUpgrade: DownloadingUpgrade: 51
ExecuteSoftwareUpgrade: DownloadingUpgrade: 52
```

The RCP host checks for software upgrades over the Internet using the HTTP protocol, and also downloads the software image using HTTP. Any errors encountered in the HTTP transaction are returned to the RCP client with the

ErrorHTTPResult token, to which is added the actual decimal HTTP response code:

```
ExecuteSoftwareUpgrade: ErrorHTTPResult: 404
```

When parsing the results of the ExecuteSoftwareUpgrade command, you can rely on the fact that any of the progress responses listed above, with the exception of UpgradeComplete, will be followed by an additional response; either another progress response or an error response. Error responses will always be the final response of this command (excepting the required TransactionComplete token).

Preconditions for successful software upgrades:

In order for a software upgrade to succeed, the unit must not be connected to a music server. Also, there can only be one software upgrade executing at a time, so if another RCP session is attempting a software upgrade (or the user is upgrading using the host-generate UI) then ExecuteSoftwareUpgrade will fail. Also, the initial setup process must be complete, and if this is a network software upgrade, the network must be active and connected to the Internet.

If a software upgrade succeeds, then the unit should be rebooted at the earliest convenience. Several functions will not be available after a successful upgrade until the unit is rebooted, and some RCP commands will return ErrorAwaitingPostUpgradeReboot in this state.

IMPORTANT:
If a software upgrade fails, additional attempts may be made. If the upgrade fails while erasing, writing, or verifying the flash ROM, then the RCP host's main run-time software image is probably corrupted and will not boot on subsequent power-ups or reboots! The user should be urged to re-attempt the software upgrades in this case so that the main flash ROM software image can be restored.

If the RCP host boots up with a corrupted run-time software image, it will boot into Safe-Mode. With current retail hardware configurations, Safe-Mode does NOT contain RCP support, but the user is able to perform the software upgrade process manually using the remote. A new version of Safe-Mode in development contains RCP support with enough commands to complete software upgrade.

Canceling a software upgrade:

The RCP client can cancel the ExecuteSoftwareUpgrade transaction, however, doing this is strongly discouraged, because the results of a software upgrade attempt should be of utmost importance to the RCP client. If the transaction is cancelled before the step in which the flash ROM is erased, then the upgrade

process is abandoned and the existing software will be left intact. However, if the upgrade process has started erasing the flash ROM, then the upgrade process will continue in the background even after the transaction has been cancelled, although no further progress or error results will be delivered to the RCP session.

Remember that even if you decide to cancel the upgrade transaction, it may be difficult to ensure that you are doing so before the flash ROM erase step due to the notification delay inherent in the communications protocol you may be using between the RCP client and the RCP host.

ResetToFactoryDefaults

Syntax:

ResetToFactoryDefaults

Classification:

Synchronous

Result Type:

No results

Responses:

Complete
NotComplete

Example:

ResetToFactoryDefaults
ResetToFactoryDefaults: Complete

ResetToFactoryDefaults is not yet implemented.

Reboot

Syntax:

Reboot

Classification:

Synchronous

Result Type:

No results

Responses:

OK

Example:

Reboot
Reboot: OK

The Reboot command reboots the RCP host. The current RCP session is terminated for rebooting and must be re-initiated by the client upon reboot.

GetFriendlyName

Syntax:

GetFriendlyName

Classification:

Synchronous

Result Type:

Returns a name string

Responses:

No responses

Example:

GetFriendlyName
GetFriendlyName: Kitchen SoundBridge

GetFriendlyName returns the current “friendly name” of the RCP host. The friendly name is a localized name that can be given by the user that uniquely identifies the device. The default friendly name is the product name string as determined by the RCP host software.

SetFriendlyName

Syntax:

SetFriendlyName name

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no name given
OK – indicates success

Example:

SetFriendlyName Dan's SoundBridge
SetFriendlyName: OK

SetFriendlyName sets the friendly name on the RCP host. The new name will be used on the host-generated web page, host DHCP requests, UPnP broadcasts, and Bonjour (formerly Rendezvous) broadcasts, following a subsequent reboot of the host. The friendly name is a localized name that can be given by the user that uniquely identifies the device.

GetOption

Syntax:

GetOption name

Classification:

Synchronous

Result Type:

Returns result token based on option parameter.

Responses:

ParameterError – invalid option name given
GenericError – internal error
ErrorUnsupported – option not supported on host

Example:

GetOption restartMode
GetOption: lastState

Use GetOption and SetOption to query and set the value of various options on the RCP host. See description of SetOption for a full listing of all option names and values.

NOTE: As of this writing, only the outputMultichannel option will return ErrorUnsupported. It will return this error if it is used on a unit that does not have digital audio out capability.

SetOption

Syntax:

SetOption name value

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid option name or value
ErrorUnsupported – option not supported on host
OK – indicates success

Example:

SetOption restartMode standby
SetOption: OK

Use GetOption and SetOption to query and set the value of various options on the RCP host. Once set, an option is persisted to flash ROM (after a short delay) and will survive subsequent reboots and power cycles. These options will revert to their default values after a Reset to Factory Defaults, however

NOTE: As of this writing, only the outputMultichannel option will return ErrorUnsupported. It will return this error if it is used on a unit that does not have digital audio out capability.

The following table lists all valid option names and values.

Option Name	Option Values
bootMode	lastState standby lastSource serverList
standbyMode	clock screenOff
outputMultichannel	Integer value 0 or 1
revertToNowPlaying	Integer value 0 or 1
scrollLongInfo	Integer value 0 or 1
displayComposer	Integer value 0 or 1
skipUnchecked	Integer value 0 or 1
wmaThreshold	Integer value 0 - 400

bootMode

The Boot Mode option controls the state into which the RCP host starts upon booting up, either from application of unit power or from a software-controlled reboot. In *lastState* mode, the unit will either return to standby mode or attempt to re-connect to the last-played music source or server, depending on the last state of the unit. If the last source was a radio station (Internet or AM/FM), then the unit will attempt to re-connect to that station and begin playback immediately. This is the default unit setting. In *standby* mode, the unit will boot into standby. In *lastSource* mode, the unit will never boot into standby mode, and will re-connect to the last music source or server. In *serverList* mode, the unit will boot to the main server list menu.

standbyMode

This option controls the behavior of the RCP-host generated UI when the user puts the unit into standby mode. The options are either to display the clock or to blank the screen and turn the display off. The default value is to display the clock.

outputMultichannel

This option is only available on units with digital audio outputs. If set to 0, the SPIDF validity bit is set on the digital audio output bitstream, otherwise, the validity bit is cleared. When playing an uncompressed multi-channel audio file with the validity bit set, some receivers incorrectly interpret the incoming signal as 2-channel PCM data. This option exists so that users with such receivers can successfully play multi-channel audio. The default value is 0.

revertToNowPlaying

This option controls the behavior of the RCP-host generated UI when playing music. If this option is set to a value of 1, then the unit will return to the Now Playing screen after a timeout of 60 seconds if a user has a menu on screen but does not hit any keys. The default value is 1.

scrollLongInfo

This option controls the behavior of the RCP-host generated UI when playing music. If set to 1, the unit will scroll long metadata strings on the Now Playing screen. Some users find this movement distracting, so this option allows them to disable scrolling. The default value is 1.

displayComposer

This option controls the behavior of the RCP-host generated UI. If the unit is playing a song in the Classical genre, then this option will substitute the composer name (“Bach”, e.g.) for the artist name (often the name of the

performing symphony, e.g.) on the Now Playing screen display. The default value is 0.

skipUnchecked

This option controls the behavior of the player when connected to a DAAP server (like iTunes). If set to 1, then the player will skip over tracks that have been "unchecked" in the server interface. (These are usually songs on an album that the user never wants to hear again.) The default value is 0.

wmaThreshold

The WMA threshold option controls the way the RCP host requests song data for WMA files from a music server. Initial versions of Windows Media Connect did not allow player clients to distinguish between WMA and WMA Lossless, so SoundBridge units use a default bitrate value of 400 kilobits per second as a metric to decide if a WMA file is encoded with WMA Lossless. (WMA files with a bitrate above 400 kbps are assumed to be encoded with WMA Lossless.)

When SoundBridge determines a song is in WMA Lossless, it requests that the server transcode the file into uncompressed PCM format for streaming. However, some WMA Lossless files (especially those with long quiet passages) are able to encode at bitrates less than 400 kbps, which makes SoundBridge unable to play them with its default settings. With this option users can instruct SoundBridge to request the server to transcode WMA files with a lower bitrate into PCM as well, in order to make those WMA Lossless files playable.

Note that the units "kilobits per second" refers to 1000 bits per second (NOT 1024 bits per second).

Display Control Commands

These commands interact with the display subsystem:

- GetVisualizer
- SetVisualizer
- VisualizerMode (DEPRECATED)
- GetVisualizerMode
- SetVisualizerMode
- ListVisualizers
- GetVizDataVU
- GetVizDataFreq
- GetVizDataScope
- DisplayUpdateEventSubscribe
- DisplayUpdateEventUnsubscribe
- GetDisplayData

These commands are described in the pages that follow.

GetVisualizer

Syntax:

GetVisualizer [verbose]

Classification:

Synchronous

Result Type:

Returns visualizer id on success.

Responses:

ErrorUnsupported – unsupported on this host

Example:

GetVisualizer
GetVisualizer: Clock

GetVisualizer returns the current active visualizer id. This identifier will be one of the items returned by ListVisualizers. Some RCP hosts do not support visualizers in the host-generated UI, and will return ErrorUnsupported.

Pass the optional "verbose" parameter to request a localized friendly name for the visualizer appropriate for displaying to the end user. This name will be translated into the current language selection.

HISTORICAL NOTE: This command was added in version 2.6.19. (TODO: note when merged into 2.7.)

SetVisualizer

Syntax:

SetVisualizer n|id|name

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid parameter
ErrorUpsupported – unsupported on this host
OK – indicates success

Example:

SetVisualizer Oscilloscope
SetVisualizer: OK

SetVisualizer sets the current active visualizer as displayed on the Now Playing screen in the UI generated by the RCP host. The visualizer can be identified either by id, localized name, or by a zero-based index into the list results of the ListVisualizers command. Some RCP hosts do not support visualizers in the host-generated UI, and will return ErrorUnsupported.

HISTORICAL NOTE: ErrorUnsupported was added in software version 2.6.19.

VisualizerMode

Syntax:

VisualizerMode full|partial|off

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid parameter
ErrorUpsupported – unsupported on this host
OK – indicates success

Example:

VisualizerMode full
VisualizerMode: OK

NOTE: VisualizerMode is deprecated! Use SetVisualizerMode instead.

GetVisualizerMode

Syntax:

GetVisualizerMode

Classification:

Synchronous

Result Type:

Returns a mode token.

Responses:

GenericError – internal error
ErrorUpsupported – unsupported on this host

Example:

GetVisualizerMode
GetVisualizerMode: full

GetVisualizerMode returns the current visualizer mode for the built-in visualization system in the host-generated UI. Some RCP hosts do not support visualizers in the host-generated UI, and will return ErrorUnsupported.

HISTORICAL NOTE: GetVisualizerMode was added to RCP in software version 2.6.19.

SetVisualizerMode

Syntax:

SetVisualizerMode full|partial|off

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid parameter
ErrorUpsupported – unsupported on this host
OK – indicates success

Example:

SetVisualizerMode full
SetVisualizerMode: OK

SetVisualizerMode updates the mode of the visualization system built-into the host-generated UI. Some RCP hosts do not support visualizers in the host-generated UI, and will return ErrorUnsupported.

HISTORICAL NOTE: GetVisualizerMode was added to RCP in software version 2.6.19.

ListVisualizers

Syntax:

ListVisualizers [verbose]

Classification:

Synchronous

Result Type:

Returns a list of visualizer names or ids. This list result can be used with the SetVisualizer command.

Responses:

ErrorUpsupported – unsupported on this host
OK – indicates success (see issue below)

Example:

ListVisualizers
ListVisualizers: ListResultSize 5
ListVisualizers: Horizontal VU-Meter
ListVisualizers: Vertical VU-Meter
ListVisualizers: Voice Print
ListVisualizers: Frequency Analyzer
ListVisualizers: Oscilloscope
ListVisualizers: ListResultEnd

ListVisualizers lists all visualizers supported by the current software version. The returned values can be used by the SetVisualizer command to set the current visualizer. Some RCP hosts do not support visualizers in the host-generated UI, and will return ErrorUnsupported.

ListVisualizers returns a list of visualizer ids by default. If you pass the optional "verbose" parameter, it will return a list of localized visualizer names, translated into the current language selection.

HISTORICAL NOTE: Until software version 2.6.19, ListVisualizers returned an additional OK result token after returning the list results.

GetVizDataVU

Syntax:

GetVizDataVU

Classification:

Synchronous

Result Type:

Returns visualizer byte data

Responses:

OK – indicates success

Example:

GetVizDataVU
GetVizDataVU: data bytes: 2
25e3
GetVizDataVU: OK

GetVizDataVU reports the maximum signal level of the left and right audio samples as measured over a small (sample rate dependent) time period. This is for use in programming a custom visualizer. The first byte of data represents the level of the left channel, and the second byte represents the right channel. The bytes are unsigned values on the range 0 – 255.

This command only returns data if there is a song currently being played.

The data result is returned as a stream of hex digits or as binary data. See SetDataResultType.

GetVizDataFreq

Syntax:

GetVizDataFreq

Classification:

Synchronous

Result Type:

Returns visualizer byte data

Responses:

OK – indicates success

Example:

GetVizDataFreq
GetVizDataFreq: data bytes: 16
0090b263d2aaf3d282153431f0b03030
GetVizDataFreq: OK

GetVizDataFreq reports the frequency response of the audio currently playing. The frequency data is in 16 bins and is representative of both audio channels collectively (i.e., it is mono data), and the frequency ranges of the bins is dependent on the sample rate of the audio. Each frequency value is encoded in a single unsigned byte on the range 0 – 255.

This command only returns data if there is a song currently being played.

GetVizDataScope

Syntax:

GetVizDataScope

Classification:

Synchronous

Result Type:

Returns visualizer byte data

Responses:

OK – indicates success

Example:

```
GetVizDataScope
GetVizDataScope: data bytes: 128
0767c72888a868c757e616c5b5557585c55607170786c627e
7282828e7e79868888828383818b777f68686e6375747e666
66e5956506a647b7f7982999c9a9896939b858189727f6472
74747b6c657a7c7f7182878d859f9ba9abada0b7bab0c2cbb
ea7a3a9a8a7a4af9d9e9f95a2a4a5a3a9aea0baa3a2a3aba3
b8a9989f9ba
GetVizDataScope: OK
```

GetVizDataScope reports the data that the WMM would use to render the ocilliscope visuzalizer. This is actual sample data rendered over a small (sample rate depenent) period. This is for use in programming a custom visualizer.

The data returned is 128 unsigned bytes on the range 0 – 255, each representing a waveform value centered at 128.

This command only returns data if there is a song currently being played.

DisplayUpdateEventSubscribe

DisplayUpdateEventUnsubscribe

Syntax:

DisplayUpdateEventSubscribe
DisplayUpdateEventUnsubscribe

Classification:

Subscription

Result Type:

No results

Responses:

ErrorAlreadySubscribed – requested subscription but already subscribed
ErrorNotSubscribed – requested unsubscription but not currently subscribed
OK – indicates success

Example:

DisplayUpdateEventSubscribe
DisplayUpdateEventSubscribe: OK
DisplayUpdateEvent: 682
DisplayUpdateEventUnsubscribe
DisplayUpdateEventUnsubscribe: OK

DisplayUpdateEventSubscribe turns on a mode in which changes in the virtual display are reported to the RCP session. For example, if the user uses the remote control to change menus, the virtual display is updated, and that fact is reported as an update event. The number following the update event is the number of changes that the display has encountered since boot, and is for informational purposes only. The GetDisplayData command should be used to discover exactly how the virtual display has changed.

Once a DisplayUpdateEvent has been sent to the RCP client, no further DisplayUpdateEvents are sent until the RCP client calls GetDisplayData, even if the display continues to change. Once the RCP client does call GetDisplayData, a DisplayUpdateEvent is sent the very *next* time the display is changed.

NOTE: The RCP host does NOT send an initial DisplayUpdateEvent on subscription, but it instead waits for the display to change first.

GetDisplayData

Syntax:

GetDisplayData

Classification:

Synchronous

Result Type:

Returns textual data when the display is in character mode, returns binary image data when the display is in bitmap mode

Responses:

No responses

Example (character mode display):

GetDisplayData
GetDisplayData: New software 2.6.7 is available
GetDisplayData: Perform upgrade? Yes > No <

Example (bitmap mode display):

GetDisplayData
GetDisplayData: data bytes: 1120
00000000000000003c0000003c0000003c0000003c00000
8383818b777f68686e6375747e66666e5956506a647b7f79
... (omitted) ...
2999c9a9896939b858189727f647274747b6c657a7c7f718
2878d859f9ba9abada0b7bab0c2cbbea7a3a9a8a7a4af9d9
e9f95a2a4a5a3a9aea0baa3a2a3aba3b8a9989f9ba

GetDisplayData returns the current contents of the RCP host-generated display. When the RCP host is a SoundBridge unit with its own display, the data returned is an exact copy of the current contents of the display. When the RCP host is a WMM, the data returned is the data contained in the current virtual display. Currently, this data represents a virtual character mode display consisting of 2 lines of 40 characters.

NOTE: The SoundBridge M500 currently does not support this command.

ISSUE:
There is no current way via RCP to determine the format of the data returned by GetDisplayData without just calling it and dynamically sensing the data return type. As an alternative, applications can connect to the unit's shell console (via a telnet/TCP connection to port 4444) and call the 'displaytype' shell command. The following table indicates the types and dimensions of each type based on the return value of the displaytype shell command:

Displaytype result	Type	Dimensions

LCD	Character	40 x 2
String	Character	40 x 2
Small VFD	Bitmap	280 x 16
Small VFD Direct	Bitmap	280 x 16
Small and Tall VFD Direct	Bitmap	280 x 32
Large VFD	Bitmap	512 x 32

Format of the Bitmap Display Data

The bitmap display data is returned as a stream of bytes. The format of the data is unusual in that the bytes represent columns of pixels. (Most bitmaps store data as bytes of pixel rows.) For example, when the display is a 280 x 16 bitmapped display, GetDisplayData will return 560 bytes of data, with every 2 bytes representing a single column of pixel data, starting at the left-most column. Within each byte, the most-significant bit represents the pixel nearest the top of the bitmap, and the least-significant bit represents the pixel nearest the bottom of the bitmap.

IR Demod/Dispatch

The host-generated user interface responds to IR events generated by the end-user. These IR events correspond to buttons that the user presses on the remote control or hard-buttons that are on the device itself, like the buttons on the top panel of the SoundBridge Radio.

When the SoundBridge / WMM user interface module processes IR commands, it updates its internal user interface display. On SoundBridge models, the effect of dispatching IR commands can be seen immediately on the physical display. On WMM, the virtual display will be updated automatically, and the latest state of the display can be queried from RCP using the GetDisplayData command.

An RCP client can dispatch arbitrary IR command codes using IrDispatchCommand, and can intercept IR commands by subscribing to IrDemod events. By intercepting IR commands, an RCP client can perform special actions on user key events and keep the RCP host-generated UI from reacting to them.

- IrDispatchCommand
- IrDemodSubscribe
- IrDemodUnsubscribe

These commands are described in the pages that follow.

IrDispatchCommand

Syntax:

IrDispatchCommand command-id|number

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError
OK

Example:

IrDispatchCommand CK_PAUSE

IrDispatchCommand injects the command argument into the RCP host's command processor, which will treat that command as if it were received from the IR remote decoder. The valid commands include:

```
CK_NORTH
CK_SOUTH
CK_WEST
CK_EAST
CK_PLAY
CK_PAUSE
CK_PLAYPAUSE
CK_STOP
CK_NEXT
CK_PREVIOUS
CK_MENU
CK_INFO
CK_EXIT
CK_POWER
CK_POWER_ON
CK_POWER_OFF
CK_SEARCH
CK_ADD
CK_SHUFFLE
CK_REPEAT
CK_VOLUME_UP
CK_VOLUME_DOWN
CK_BRIGHTNESS
CK_ROTARY_CLOCKWISE
CK_ROTARY_COUNTERCLOCKWISE
CK_ROTARY_SWITCH
CK_SNOOZE
CK_SOURCE
CK_SCAN_UP
CK_SCAN_DOWN
CK_GROUP
CK_ALARM
CK_PLAYLISTS
CK_BROWSE_ARTISTS
CK_BROWSE_ALBUMS
CK_BROWSE_SONGS
CK_BROWSE_GENRES
CK_BROWSE_COMPOSERS
CK_PRESET_A1
CK_PRESET_A2
CK_PRESET_A3
CK_PRESET_A4
CK_PRESET_A5
CK_PRESET_A6
CK_PRESET_B1
CK_PRESET_B2
CK_PRESET_B3
CK_PRESET_B4
CK_PRESET_B5
CK_PRESET_B6
CK_PRESET_C1
CK_PRESET_C2
CK_PRESET_C3
CK_PRESET_C4
CK_PRESET_C5
CK_PRESET_C6
CK_INTERNET_RADIO
CK_FM_RADIO
CK_AM_RADIO
CK_LAST_MUSIC_SERVER
CK_VOLUME_50
```

IrDemodSubscribe

IrDemodUnsubscribe

Syntax:

IrDemodSubscribe [updown]
IrDemodUnsubscribe

Classification:

Subscription

Result Type:

No results

Responses:

ErrorAlreadySubscribed – requested subscription but already subscribed
ErrorNotSubscribed – requested unsubscription but not currently subscribed
OK – indicates success

Example:

IrDemodSubscribe
IrDemodSubscribe: OK
IrDemod: CK_PAUSE
IrDemod: CK_PREVIOUS
IrDemodUnsubscribe
IrDemodUnsubscribe: OK

IrDemodSubscribe disconnects the stream of IR commands sent from the RCP host's IR decoder to the command processor, and sends them to the RCP session instead. When the IR decoder receives an IR signal, it will decode it and the appropriate command will be delivered to the client over the RCP session.

IrDemodUnsubscribe reconnects the stream of IR commands sent from the RCP host's IR decoder to the command processor, and they will no longer be echoed to the RCP session.

Listing and Connecting to Media Servers

Overview

- ListServers
- SetServerFilter
- SetServerConnectPassword
- ServerConnect
- ServerLaunchUI
- ServerDisconnect
- GetConnectedServer
- GetActiveServerInfo
- ServerGetCapabilities

The commands in this section are used for listing, connecting to, and disconnecting from music servers in the RCP session, and for getting information about the connected server. The RCP host automatically detects media servers on the local-area network that advertise using the UPnP/AV, Bonjour (formerly Rendezvous), and SlimServer protocols. Additional servers are added automatically to the list for accessing Internet Radio and other media devices depending on the unit's configuration. (E.g., the SoundBridge Radio adds servers for AM and FM radio.) RCP clients may specify what types of media servers to list by using the SetServerFilter command.

RCP Session Active Server
Each RCP session has an *active server*, which you can visualize as a per-RCP session member variable that can be pointed to one of the currently discovered music servers. When a new RCP session begins, the active server is empty, and must be initialized before use.

The design of the per-RCP session active server paradigm was motivated by the eventual goal of allowing SoundBridge and WMM to connect to multiple music servers at the same time, such that the user could actually create a playlist that contained some songs from server A and some songs from server B. However, the current implementation only allows the SoundBridge / WMM to connect to one music server at a time, so some of the server-related RCP commands may seem a little unwieldy.

The situation is further complicated by the fact that other clients outside of the control of the RCP session can initiate server connections. E.g., a user using the remote control can connect a SoundBridge unit to a music server before the RCP session begins. Or, depending its user-settings, the SoundBridge / WMM might automatically connect to a music server on bootup. (The RCP active server is NOT initialized to point to this server automatically!)

If the unit is connected to a music server outside of the RCP session, the RCP session must use the GetConnectedServer command to "attach" its active server

to the server that the unit has been connected to. Once the session has attached its active server thusly, it can access that server's content with commands like ListSongs, or it can disconnect using ServerDisconnect and then connect some other music server.

To connect to a media server, the RCP client must first use ListServers to get a list of currently-available servers. (Use the SetServerFilter command to limit the list of servers returned to a certain type, like just UPnP servers or just DAAP (iTunes) servers.) Then the RCP client can use either ServerConnect or ServerLaunchUI to connect to one of these servers. Use ServerConnect when you want to handle all the possible connection errors yourself, like requiring a password. Use ServerLaunchUI when you want the SoundBridge / WMM to generate UI screens for the user to resolve any connection problems. (If a password is required, e.g., ServerLaunchUI will put the password-entry screen up on the UI and wait for remote key presses from the user to complete the connection.)

All of the content browsing, searching, and queuing commands require a valid active server, so understanding these server connection commands is vital for full utilization of RCP.

ListServers

Syntax:

ListServers

Classification:

Synchronous, returns list results

Result Type:

Returns a list result of media servers on success. This list result can be used with the following commands:

- **ServerConnect**
- **ServerLaunchUI**

Responses:

ErrorInitialSetupRequried – initial setup not complete
GenericError – indicates internal error

Example:

ListServers
ListServers: ListResultSize 3
ListServers: Some Server Name
ListServers: Some other server name
ListServers: Internet Radio
ListServers: ListResultEnd

The “ListServers” command takes no arguments, and returns the list of media servers discovered on the local network, plus a server representing Internet Radio stations, plus optional servers that represent physical devices (like AM/FM tuner, line-in, or memory cards) connected to the unit. The list is sorted alphabetically by name for all servers except the Internet Radio server, AM / FM Radio servers, and Line-In server, which are added at the end of the list when available.

The list will contain servers of the types indicated by the current server filter, which can be set with SetServerFilter. By default, all server types are listed.

The string value in the list results is the “friendly name” that each server publishes for itself. The ListServers command returns the entire friendly name as delivered by the UPnP and Rendezvous protocols. Also note that the server names for the Internet Radio server and AM / FM Tuner servers are localized depending on the currently active language selection.

The WMM automatically searches the local network for media servers in the background, so the server listing is a synchronous command and will immediately return the current list of servers detected.

The side-effect of this command is that it returns list results, clobbering any existing list results held by the RCP session, which can be used by subsequent invocations of ServerConnect or ServerLaucnhUI. The scope of this command is restricted to the RCP session.

NOTE: ErrorInitialSetupRequired was added in software version 2.6.XXX.

SetServerFilter

Syntax:

SetServerFilter <filter-tokens>

Classification:

Synchronous

Result Type:

No results

Error Responses:

ParameterError – invalid server filter
OK – indicates success

Example:

SetServerFilter daap upnp
SetServerFilter: OK

Use SetServerFilter to set which types of media servers should be returned by the command ListServers. The parameters to SetServerFilter are a space-delimited list of the following tokens (capitalization is ignored):

- "daap" – servers using iTunes DAAP protocol
- "upnp" – servers using UPnP/AV protocol (e.g., Windows Media Connect, Rhapsody, MusicMatch, etc.)
- "rsp" – Firefly music servers (see www.rokulabs.com/firefly)
- "slim" – open-source SlimServer product by SlimDevices
- "radio" – pseudo-server for connecting directly to Internet Radio
- "flash" – directly connected devices, e.g., a flash card physically inserted into a slot on the device
- "linein" – some WMM devices have a direct audio input
- "am" – AM Tuner on the SoundBridge Radio
- "fm" – FM Tuner on the SoundBridge Radio
- "all" – list all server types
- "debug" – list debug server discovery info (see below)

The side-effect of this command is to set the per-RCP session server filter, which affects the execution of the ListServers command. The default value of the per-RCP session server filter is "all".

The "debug" server filter option causes the ListServers command to list additional information about all discovered music servers. Each line of output from the ListServers command will contain an availability token, a server type token, and finally the server name, all separated by a single space character.

The availability token can have one of the following values: kOnline, kOffline, kHidden, and kInaccessible. The kHidden state indicates that the server has another service that is taking priority over this one. (E.g., Firefly servers publish an RSP service and a DAAP service, and the SoundBridge prefers to connect to the RSP one, thereby giving the DAAP service the kHidden attribute.) The kInaccessible state indicates that the RCP host configuration will not allow connection to this server.

The server type token indicates the type of each server, and has one of the following values: kITunes, kUPnP, kSlim, kFlash, kFavoriteRadio, kAMTuner, kFMTuner, kRSP, and kLineIn.

IMPORTANT: The “debug” server filter and its operation is not an officially supported feature of RCP, so robust RCP clients should NOT rely on its operation. Roku may change or remove this functionality in the future.

NOTE: The browsing, searching and queuing RCP functions do not work with slim servers, so the only use for connecting to a slim server in RCP is to use the RCP host-generated UI to access it.

HISTORICAL NOTE: Until software version 2.5.155, the RCP server filter was global across all RCP sessions.

SetServerConnectPassword

Syntax:

SetServerConnectPassword [password]

Classification:

Synchronous

Result Type:

No results

Responses:

OK

Example:

SetServerConnectPassword secret
SetServerConnectPassword: OK

SetServerConnectPassword sets the per-RCP session server connect password variable to the password parameter. The password can be any string of text. If the password parameter is not given, then the server connect password variable is cleared.

The server connect password is used by the ServerConnect command when connecting to a media server. An RCP client can tell if a server requires a connection password by calling ServerConnect and receiving the ConnectionFailedPassword error result. This result is also sent if the password is incorrect.

Note that the password to a password-protected media server only needs to be provided once during the boot session of the RCP host. The server can be connected and disconnected subsequently without having to re-supply the password.

ServerConnect

Syntax:

ServerConnect n

Classification:

Transacted

Result Type:

No results

Responses:

ParameterError – invalid index or list results
ErrorUpgradeInProgress – a software upgrade is in progress
ErrorAwaitingPostUpgradeReboot – a software upgrade has completed and unit needs reboot
ResourceAllocationError – another user of the system is currently connecting or disconnecting from a server, wait and try again
ConnectionFailedAlreadyConnected – the active server is currently set, or another user is currently connected already (see below)
ConnectionFailedNoNetwork – this server requires an active net connection
ConnectionFailedBusy – server refused connection
ConnectionFailedPassword – server requires password or current password is incorrect
ConnectionFailedForbidden – server refused connection
ConnectionFailedWMCUnauthorized – device requires server authorization
ConnectionFailedNoContact – no response from server
ConnectionFailedUnknown – other failure
GenericError – other failure, usually caused by another user connecting to a server simultaneously
Connected – indicates success

Example:

ServerConnect 0
ServerConnect: TransactionInitiated
ServerConnect: Connected
ServerConnect: TransactionComplete

ServerConnect will connect to a music server, setting the RCP session's active server on success. To use this function, you must first call ListServers to set the current per-RCP session list result to a list of available servers. Then execute ServerConnect with a numeric parameter that is the zero-based index into the list returned by ListServers.

ServerConnect is a transacted command since it requires network access to contact the requested server and to perform any steps required before it is usable. Note that there are some error conditions that can be sensed immediately upon execution, so if RCP may respond with an error code instead of the TransactionInitiated token, in which case there is no transaction and no further results will be sent.

Before connecting to a server, the RCP session must be disconnected from any other server. Furthermore, another user (another RCP session or a user operating the unit with a remote control or from hard-buttons or the web page) could be connected to a server, which would also result in a connection error.

To make sure that no one in the system is connected to a server, first call GetConnectedServer, and then call ServerDisconnect.

If set, ServerConnect sends the per-RCP session server password to the server as a log-in password. This password can be set using the SetServerConnectPassword command. If the server password is incorrect or the server requires a password and none is set, the result will be ConnectFailedPassword. If ServerConnect initiates a transaction, then the per-RCP session server password will be cleared. (Note that the correct server password only needs to be provided once in the boot session of the RCP host device; subsequent connects to the same server will use the cached server password.)

Each time in the RCP session that a ServerConnect succeeds, the per-RCP session current container path variable is reset to the root. This is only applicable to servers that expose a container hierarchy. See the container commands GetCurrentContainerPath, ListContainerContents, ContainerEnter, and ContainerExit.

Many RCP commands require that the RCP session have a connected active server, including all of the song and category listing and searching commands and the song queuing commands.

Also note that ServerLaunchUI can also be used to connect to a music server, but this command might cause the RCP host-generated UI to block waiting for user input if the server requires a password, and it will wait for several seconds displaying an error message if the connection fails.

ServerLaunchUI

Syntax:

ServerLaunchUI n

Classification:

Transacted

Result Type:

No results

Responses:

ParameterError – invalid index or list results
ErrorUpgradeInProgress – a software upgrade is in progress
ErrorAwaitingPostUpgradeReboot – a software upgrade has completed and unit needs reboot
ResourceAllocationError – another user of the system is currently connecting or disconnecting from a server, wait and try again
ConnectionFailedAlreadyConnected – the active server is currently set, or another user is currently connected already (see below)
GenericError – other failure, usually caused by another user connecting to a server simultaneously
Connected – indicates success

Example:

ServerLaunchUI 0
ServerLaunchUI: TransactionInitiated
ServerLaunchUI: Connected
ServerLaunchUI: TransactionComplete

Like ServerConnect, ServerLaunchUI will connect to a currently-discovered media server. All of the prerequisites of ServerConnect apply here as well, like the need for all users of the device to be currently disconnected from a server. The parameter is a zero-based index into a server list result from the ListServers command.

ServerLaunchUI differs from ServerConnect in how it handles errors during the connection process. If a connection initiated with ServerLaunchUI encounters any errors during the connection, the RCP host device will generate error screens in its built-in UI (exposed via the GetDisplayData RCP command and shown on the device's display if it has one). One important connection error requiring user interaction is if the server requires a password and the correct password has not been entered yet during the device's boot session. In this

case, the host device will generate a password entry screen on its display and wait for user input to complete the connection process.

Also, ServerLaunchUI does NOT use the per-RCP session server connect password set by the SetServerConnectPassword command.

ServerLaunchUI will reset the per-RCP session container path to the root upon successful connection.

ServerDisconnect

Syntax:

ServerDisconnect

Classification:

Transacted

Result Type:

No results

Responses:

ErrorDisconnected – the active server is not set
ResourceAllocationError – another user is currently connecting or disconnecting
GenericError – an unexpected error occurred
Disconnected – indicates success

Example:

ServerDisconnect
ServerDisconnect: TransactionInitiated
ServerDisconnect: Disconnected
ServerDisconnect: TransactionComplete

ServerDisconnect will disconnect the RCP host from the RCP session's active server. A server must be disconnected before the RCP host will be able to connect to another server.

Note that if the RCP host has been connected to a server by some other mechanism than a the RCP session, then the active server will not be set to that server yet, and the RCP client must call GetConnectedServer before it will be able to disconnect from it!

Disconnecting from a music server will affect other active RCP sessions on the RCP host, and it will affect the user interacting with the SoundBridge using the remote control! Other RCP sessions will notice that their calls to any RCP function that requires the connected server will fail, and they would have to call GetConnectedServer to re-sync their active server to the newly connected one, if there is one, or they will have to call ServerDisconnect to clear out their active server before attempt to re-connect to another one.

Note that in the current software release, if the current server connection was initiated either by ServerLaunchUI or by a user using the remote control, then the RCP host will generate a disconnection message and display it for 5 seconds during the server disconnect process to notify the interactive user that the server

connection has been cut. (The ServerDisconnect transaction will wait this additional 5 seconds as well before completing.)

GetConnectedServer

Syntax:

ServerDisconnect

Classification:

Synchronous

Result Type:

No results

Responses:

GenericError – there is no currently connected server
OK – indicates success

Example:

GetConnectedServer
GetConnectedServer: OK

GetConnectedServer will sync-up the per-RCP session active server with the currently connected server on the RCP host.

This command is necessary when the RCP host gets connected to a server by some mechanism other than the current RCP session's use of the ServerConnect or ServerLaunchUI commands. E.g., a user with a remote control can connect to a music server. (This scenario might not apply for a WMM that does not have an IR receiver.) Or, a user can launch a preset from a hard-button on the unit or from the web page, or another RCP session can connect to another server. OR, the unit might auto-connect to a server on bootup.

All these scenarios turn the RCP session's active server into an orphan; it ends up referencing a server that is no longer connected. (Or, if the RCP session has just been initiated, it will not have been set yet.)

Here are suggested times that it is a good idea to call GetConnectedServer:

- Upon the initial connection of the RCP session – if it returns OK, then you know that the RCP host is already connected to a server and you can immediately start using it (or disconnect from it!)
- If ServerConnect or ServerLaunchUI returns ConnectionFailedAlreadyConnected
- If any of the RCP commands that access a music server start failing. (They currently return GenericError in this instance; in future software releases this might change to ErrorDisconnected.)

NOTE: One scenario is particularly tricky: if your RCP session has an active server but some other user disconnects from that server and does NOT reconnect to another server, then GetConnectedServer will return GenericError, but it will NOT clear out your active server. You will still have to call ServerDisconnect to clear it out and allow future server connections from that RCP session. (ServerDisconnect will initiate a transaction and return ErorDisconnected, but it will also clear out the active server.)

BUG: When GetConnectedServer syncs up the RCP session's active server to a new server, the RCP session's current container path is NOT reset to the root path. This will be addressed in a future software release.

GetActiveServerInfo

Syntax:

GetActiveServerInfo

Classification:

Synchronous

Result Type:

##########

Error Responses:

ErrorDisconnected – there is no active server
OK – indicates success

Example:

GetActiveServerInfo
GetActiveServerInfo: Type: rsp
GetActiveServerInfo: Name: Firefly Media Server
GetActiveServerInfo: OK

GetActiveServerInfo returns the name and protocol type of the active server. The protocols returned are similar to those used by the SetServerFilter command, excluding the "all" and "debug" items:

- "daap" – servers using iTunes DAAP protocol
- "upnp" – servers using UPnP/AV protocol (e.g., Windows Media Connect, Rhapsody, MusicMatch, etc.)
- "rsp" – Firefly music servers (see www.rokulabs.com/firefly)
- "slim" – open-source SlimServer product by SlimDevices
- "radio" – pseudo-server for connecting directly to Internet Radio
- "flash" – directly connected devices, e.g., a flash card physically inserted into a slot on the device
- "linein" – some WMM devices have a direct audio input
- "am" – AM Tuner on the SoundBridge Radio
- "fm" – FM Tuner on the SoundBridge Radio

ServerGetCapabilities

Syntax:

ServerGetCapabilities

Classification:

Transacted

Result Type:

No results

Error Responses:

ErrorDisconnected – this RCP session has no active server
GenericError – command failure

Example (Rhapsody):

```
ServerGetCapabilities
ServerGetCapabilities: TransactionInitiated
ServerGetCapabilities: QuerySupport: None
ServerGetCapabilities: Containers: yes
ServerGetCapabilities: Playlists: no
ServerGetCapabilities: PartialResults: yes
ServerGetCapabilities: TransactionComplete
```

Example (Firefly):

```
ServerGetCapabilities
ServerGetCapabilities: TransactionInitiated
ServerGetCapabilities: QuerySupport: Partial
ServerGetCapabilities: Containers: no
ServerGetCapabilities: Playlists: yes
ServerGetCapabilities: PartialResults: yes
ServerGetCapabilities: TransactionComplete
```

ServerGetCapabilities returns information about the current server that the RCP Client can use to tailor its UI, removing or adding options relevant to the particular server. There are two examples above – one for a UPnP server (Rhapsody), and one for an RSP server (Firefly). Note the differences in capabilities between these two servers. Both support partial results (see below), but the UPnP server supports only browsing of a container hierarchy, while the RSP server supports partial-string queries and direct access to playlists.

ServerGetCapabilities returns capabilities in four areas:

QuerySupport
The QuerySupport section of the response will contain one of the following options to indicate what types of queries are supported:

- "None" – server does not support queries of any kind
- "Songs" – client may ask for the server's list of songs, but no other queries are allowed
- "Basic" – server supports basic query features required to use the SetBrowseFilterXXXX commands
- "Partial" – server supports substring queries using the SearchXXXX commands

Containers
The Containers section of the response is either:

- "yes" – server supports browsing by container
- "no" – server does not support browsing by container

Playlists
The Playlists section of the response is either:

- "yes" – server supports direct listing of playlists
- "no" – server does not support direct listing of playlists

PartialResults
The PartialResults section indicates whether the server can accept queries limited to a range of items (e.g. only return items 0-49). At this time, RCP does not expose this functionality to the client (i.e. RCP always fetches entire result lists), but this section is included for future use.

- "yes" – server supports direct listing of playlists
- "no" – server does not support direct listing of playlists

Content Selection And Playback

Overview

The following commands allow you to list, organize, and play music stored on a server. For these commands to succeed, the RCP session must have a valid, connected active server. See the preceding section for details.

- ListSongs
- ListAlbums
- ListArtists
- ListComposers
- ListGenres
- ListLocations
- ListMediaLanguages
- ListPlaylists
- ListPlaylistSongs
- ListContainerContents
- GetCurrentContainerPath
- ContainerEnter
- ContainerExit
- SearchSongs
- SearchArtists
- SearchAlbums
- SearchComposers
- SearchAll
- SetBrowseFilterAlbum
- SetBrowseFilterArtist
- SetBrowseFilterComposer
- SetBrowseFilterGenre
- SetBrowseFilterLocation
- SetBrowseFilterMediaLanguage
- SetBrowseFilterTopStations
- SetBrowseFilterFavorites
- SetSongListSort
- SetBrowseListSort

Browsing

Music library browsing is supported with the ListXXXX commands and the SetBrowseFilterXXXX commands. Browsing means that you can search the library by categories, namely: artist, album, composer, and genre. Not all music libraries support browsing, so be prepared for a failure result. In particular, some UPnP servers don't support browsing, but require you to browse containers instead. The ServerGetCapabilities command can provide guidance as to the connected server's capabilities.

Each RCP session has a “browse filter” object that you can set up with parameters to modify the lists returned by the next call to a ListXXXX command. For example, to list the albums by the artist Led Zeppelin, you could issue the following commands:

```
SetBrowseFilterArtist Led Zeppelin
ListAlbums
```

Once the RCP client issues a ListXXXX command, the browse filter is cleared, so if you next want to drill down and list all the songs from the Led Zeppelin album Presence, you would issue the following commands:

```
SetBrowseFilterArtist Led Zeppelin
SetBrowseFilterAlbum Presence
ListSongs
```

Filter term matching is done by textual comparisons, so if there are two albums in the music library with the same name but different artists, if you just set the album filter you can still match songs in the album of the same name from the other artist.

NOTE: The Internet Radio server allows browsing using two distinct categories compared to other browse-capable servers. When connected to the Internet Radio server, the following browse categories are available: location and media language.

Searching

Many media servers also support searching. (Most servers that support browsing also support searching.) Searching basically allows you to specify a substring with which you want to find matches in one of the following categories: song, artist, album, or genre. So to search for all artists that contain the string “Bob”, you could issue the following command:

```
SearchArtists Bob
```

Browsing Containers

Some media servers (mainly UPnP servers) allow you to browse a container hierarchy, roughly equivalent to a directory structure you might find on a PC. The directories and organization is dependent on the server and varies greatly from server to server.

Each RCP session contains a container path variable that contains the current container path. The RCP session can list the contents of the current container, and then enter sub-containers and list the contents there. See the descriptions of the container commands for further details.

Sorting Results

Beginning with software version 3.0.24, there are two commands that allow control over the sorting order of items returned by the various SearchXXXX and ListXXXX commands. See the descriptions of SetBrowseListSort and SetSongListSort for detailed information.

ListSongs

Syntax:

ListSongs

Classification:

Transacted

Result Type:

Returns a list of Song objects. This song list can be used with the following commands:

- **GetSongInfo**
- **NowPlayingInsert**
- **PlayIndex**
- **QueueAndPlay**
- **QueueAndPlayOne**

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListSongs
ListSongs: TransactionInitiated
ListSongs: ListResultSize 2
ListSongs: Doctor Doctor
ListSongs: Gold Dust Woman
ListSongs: ListResultEnd
ListSongs: TransactionComplete

ListSongs lists the songs on the RCP session's active server. If any browse filters have been set, then the song listing will be restricted to songs that match the browse filter terms.

As a side effect of executing this command, the current browse filters are cleared.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListAlbums

Syntax:

ListAlbums

Classification:

Transacted

Result Type:

Returns a list of album names

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListAlbums
ListAlbums: TransactionInitiated
ListAlbums: ListResultSize 2
ListAlbums: A Love Supreme
ListAlbums: Rattle And Hum
ListAlbums: ListResultEnd
ListAlbums: TransactionComplete

ListAlbums lists the albums on the RCP session's active server. If any browse filters have been set, then the album listing will be restricted to albums that match the browse filter terms. For example, the RCP client can set the artist browse filter to an artist name before calling ListAlbums and the resulting album list will be restricted to the albums by that artist.

As a side effect of executing this command, the current browse filters are cleared.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListArtists

Syntax:

ListArtists

Classification:

Transacted

Result Type:

Returns a list of artist names

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListArtists
ListArtists: TransactionInitiated
ListArtists: ListResultSize 2
ListArtists: Aerosmith
ListArtists: ZZ Top
ListArtists: ListResultEnd
ListArtists: TransactionComplete

ListArtists lists the artists on the RCP session's active server. The genre browse filter may be used to filter the list of artists. (On some music servers, the other browse filters will also filter the results by only including artists with songs or albums that match the respective browse criteria.)

As a side effect of executing this command, the current browse filters are cleared.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListComposers

Syntax:

ListComposers

Classification:

Transacted

Result Type:

Returns a list of composer names

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListComposers
ListComposers: TransactionInitiated
ListComposers: ListResultSize 2
ListComposers: Billie Joe Armstrong/Green Day
ListComposers: The Tragically Hip
ListComposers: ListResultEnd
ListComposers: TransactionComplete

ListComposers lists the composers on the RCP session's active server. Support of browsing by composers is server-dependent, as is the effect of setting the browse filters prior to calling ListComposers.

As a side effect of executing this command, the current browse filters are cleared.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListGenres

Syntax:

ListGenres

Classification:

Transacted

Result Type:

Returns a list of genre names

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListGenres
ListGenres: TransactionInitiated
ListGenres: ListResultSize 2
ListGenres: Classical
ListGenres: Rock
ListGenres: ListResultEnd
ListGenres: TransactionComplete

ListGenres lists the genres on the RCP session's active server. Support of browsing by genres is server-dependent, as is the effect of setting the browse filters prior to calling ListGenres.

NOTE: The Internet Radio server does support browsing by genres, but it does not support filtering the list of genre results.

As a side effect of executing this command, the current browse filters are cleared.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListLocations

Syntax:

ListLocations

Classification:

Transacted

Result Type:

Returns a list of internet radio locations

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListLocations
ListLocations: TransactionInitiated
ListLocations: ListResultSize 2
ListLocations: Europe
ListLocations: USA
ListLocations: ListResultEnd
ListLocations: TransactionComplete

ListLocations lists the originating locations of internet radio stations. Only the Internet Radio server supports browsing by location. Location results can NOT be filtered using the browse filters.

As a side effect of executing this command, the current browse filters are cleared.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListMediaLanguages

Syntax:

ListMediaLanguages

Classification:

Transacted

Result Type:

Returns a list of internet radio languages

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListMediaLanguages
ListMediaLanguages: TransactionInitiated
ListMediaLanguages: ListResultSize 2
ListMediaLanguages: English
ListMediaLanguages: Spanish
ListMediaLanguages: ListResultEnd
ListMediaLanguages: TransactionComplete

ListMediaLanguages lists the languages of internet radio stations. Only the Internet Radio server supports browsing by language. Language results can NOT be filtered using the browse filters.

As a side effect of executing this command, the current browse filters are cleared.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListPlaylists

Syntax:

ListPlaylists

Classification:

Transacted

Result Type:

Returns a list of playlist objects. This list can be used with the ListPlaylistSongs command.

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListPlaylists
ListPlaylists: TransactionInitiated
ListPlaylists: ListResultSize 2
ListPlaylists: Mellow Playlist
ListPlaylists: Rockin' Out
ListPlaylists: ListResultEnd
ListPlaylists: TransactionComplete

ListPlaylists lists the playlists on the RCP session's active server. The list of playlists can NOT be filtered; i.e., the browse filters have no effect on ListPlaylists results. Not all servers support exporting playlists as independently browse-able items; often, servers that support containers only export playlists as part of the container hierarchy and not through this command.

NOTE: In order to play the songs in a playlist, the RCP client must first list the contents of the playlist using the GetPlaylistSongs command. The list of songs returned by that command can then be queued for playback.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListPlaylistSongs

Syntax:

ListPlaylistSongs n

Classification:

Transacted

Result Type:

Returns a list of Song objects. This list can be used with the following commands:

- **GetSongInfo**
- **NowPlayingInsert**
- **PlayIndex**
- **QueueAndPlay**
- **QueueAndPlayOne**

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListPlaylistSongs 0
ListPlaylistSongs: TransactionInitiated
ListPlaylistSongs: ListResultSize 2
ListPlaylistSongs: Babe I'm Gonna Leave You
ListPlaylistSongs: Cinnamon Girl
ListPlaylistSongs: ListResultEnd
ListPlaylistSongs: TransactionComplete

ListPlaylistSongs lists the songs in a playlist. The current RCP session list results must be a list of playlists returned by the ListPlaylists command. The parameter is a zero-based index of the desired playlist in that list.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

ListContainerContents

Syntax:

ListContainerContents

Classification:

Transacted

Result Type:

Returns a list of Song objects. This list can be used with the following commands:

- **GetSongInfo**
- **NowPlayingInsert**
- **PlayIndex**
- **QueueAndPlay**
- **QueueAndPlayOne**
- **ContainerEnter**

Responses:

ErrorDisconnected – this result is not currently sent, but future software versions may use it
GenericError – command failure
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

ListContainerContents
ListContainerContents: TransactionInitiated
ListContainerContents: ListResultSize 2
ListContainerContents: Music
ListContainerContents: Video
ListContainerContents: ListResultEnd
ListContainerContents: TransactionComplete

ListContainerContents lists the contents of the container described by the per-RCP session container path variable. The current container path can be reported by the GetCurrentContainerPath command.

Only some servers allow browsing the container hierarchy, but some servers *only* allow this type of content browsing! Container browsing is currently only supported by UPnP servers and flash card servers. If the server does not support containers, ListContainerContents returns GenericError.

ISSUE: The items in the list returned by ListContainerContents could be either songs or other containers, and it is not very easy to tell which is which. When using UPnP servers, it is fairly safe to assume that all the items within the same

container are either all containers or all songs. This might not be the case for flash card servers, however. The RCP client can perform a GetSongInfo on an item and check the status field – if the status is unsupported, you can assume that this is a container.

Use the ContainerEnter command to enter containers within the current container, and use ContainerExit to exit the current container and return to its parent container.

When the RCP session initially connects to a music server, the current container path is reset to the root container, i.e. "/".

BUG: When GetConnectedServer syncs up the RCP session's active server to a new server, the RCP session's current container path is NOT reset to the root path. This will be addressed in a future software release.

HISTORICAL NOTE: Until software version 2.5.155, the current container path was global across all RCP sessions.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

GetCurrentContainerPath

Syntax:

GetCurrentContainerPath

Classification:

Synchronous

Result Type:

Returns a string representing the current container path

Responses:

No Responses

Example:

GetCurrentContainerPath
GetCurrentContainerPath: /All Tracks/

GetCurrentContainerPath returns the RCP session's current container path used by the ListContainerContents command. The return result is undefined if the RCP session does not have a valid connected active server.

The container path is delimited by the forward slash character.

The container path can be modified with the ContainerEnter and ContainerExit commands.

NOTE: Not all servers publish a container hierarchy, but most UPnP servers do.

ISSUE: Containers whose names contain a forward slash are not currently escaped. This bug will be addressed in a future software release.

HISTORICAL NOTE: Until software version 2.5.155, the current container path was global across all RCP sessions.

ContainerEnter

Syntax:

ContainerEnter n

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid parameter
OK – indicates success

Example:

ContainerEnter 0
ContainerEnter: OK

ContainerEnter changes the RCP session's current container path to include the container referenced by the zero-based index parameter, which is an index into the list results returned by the GetContainerContents command. If the index parameter is out of bounds or references a non-container object, then ContainerEnter returns ParameterError. As a side-effect, when ContainerEnter succeeds, it invalidates any current list results returned by the ListContainerContents command.

NOTE: Not all servers publish a container hierarchy, but most UPnP servers do.

HISTORICAL NOTE: Until software version 2.5.155, the current container path was global across all RCP sessions.

ContainerExit

Syntax:

ContainerExit

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – already at root container
OK – indicates success

Example:

ContainerExit
ContainerExit: OK

ContainerExit changes the RCP session's current container path to its parent container. If the current container path is already at the root container, it returns ParameterError. As a side-effect, when ContainerExit succeeds, it invalidates any current list results returned by the ListContainerContents command.

NOTE: Not all servers publish a container hierarchy, but most UPnP servers do.

HISTORICAL NOTE: Until software version 2.5.155, the current container path was global across all RCP sessions.

SearchSongs

Syntax:

SearchSongs search-string

Classification:

Transacted

Result Type:

Returns a list of Song objects. This list can be used with the following commands:

- **GetSongInfo**
- **NowPlayingInsert**
- **PlayIndex**
- **QueueAndPlay**
- **QueueAndPlayOne**

Responses:

ParameterError – missing search string
GenericError – an error occurred
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

```
SearchSongs ever
SearchSongs: TransactionInitiated
SearchSongs: ListResultSize 4
SearchSongs: Tomorrow Never Knows
SearchSongs: Everything
SearchSongs: Fever Dream
SearchSongs: I'm A Believer
SearchSongs: ListResultEnd
SearchSongs: TransactionComplete
```

SearchSongs returns a list of songs on the RCP session's active server whose titles match the search string parameter. In order to be included in the results list, a contiguous substring of the song's title attribute must match the search string. The string comparison is case insensitive.

NOTE: Not all music servers support searching.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

SearchArtists

Syntax:

SearchArtists search-string

Classification:

Transacted

Result Type:

Returns a list of artist names

Responses:

ParameterError – missing search string
GenericError – an error occurred
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

SearchArtists jim
SearchArtists: TransactionInitiated
SearchArtists: ListResultSize 2
SearchArtists: Jimi Hendrix
SearchArtists: Jimmy Cliff
SearchArtists: ListResultEnd
SearchArtists: TransactionComplete

SearchArtists returns a list of artist names on the RCP session's active server that match the search string parameter. In order to be included in the results list, a contiguous substring of the artist name must match the search string. The string comparison is case insensitive.

Any of the strings returned in the results list can be used as a parameter to the SetBrowseFilterArtist command.

NOTE: Not all music servers support searching.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

SearchAlbums

Syntax:

SearchAlbums search-string

Classification:

Transacted

Result Type:

Returns a list of album names

Responses:

ParameterError – missing search string
GenericError – an error occurred
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

```
SearchAlbums ever
SearchAlbums: TransactionInitiated
SearchAlbums: ListResultSize 2
SearchAlbums: Everybody Knows This Is Nowhere
SearchAlbums: Only Everything
SearchAlbums: ListResultEnd
SearchAlbums: TransactionComplete
```

SearchAlbums returns a list of album names on the RCP session's active server that match the search string parameter. In order to be included in the results list, a contiguous substring of the album name must match the search string. The string comparison is case insensitive.

Any of the strings returned in the results list can be used as a parameter to the SetBrowseFilterAlbum command.

NOTE: Not all music servers support searching.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

SearchComposers

Syntax:

SearchComposers search-string

Classification:

Transacted

Result Type:

Returns a list of composer names

Responses:

ParameterError – missing search string
GenericError – an error occurred
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

SearchComposers bill
SearchComposers: TransactionInitiated
SearchComposers: ListResultSize 2
SearchComposers: Bill Ward/Geezer Butler
SearchComposers: Billie Joe Armstrong/Green Day
SearchComposers: ListResultEnd
SearchComposers: TransactionComplete

SearchComposers returns a list of composer names on the RCP session's active server that match the search string parameter. In order to be included in the results list, a contiguous substring of the composer name must match the search string. The string comparison is case insensitive.

Any of the strings returned in the results list can be used as a parameter to the SetBrowseFilterComposer command.

NOTE: Not all music servers support searching.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

SearchAll

Syntax:

SearchAll search-string

Classification:

Transacted

Result Type:

Returns a list of Song objects. This list can be used with the following commands:

- **GetSongInfo**
- **NowPlayingInsert**
- **PlayIndex**
- **QueueAndPlay**
- **QueueAndPlayOne**

Responses:

ParameterError – missing search string
GenericError – an error occurred
StatusSendingRequest – verbose mode only
StatusAwaitingReply – verbose mode only
StatusReceivingData – verbose mode only

Example:

SearchAll dylan
SearchAll: TransactionInitiated
SearchAll: ListResultSize 4
SearchAll: Blowin' In The Wind
SearchAll: Bob Dylan Blues
SearchAll: Like A Rolling Stone
SearchAll: One Headlight
SearchAll: ListResultEnd
SearchAll: TransactionComplete

SearchAll returns a list of songs on the RCP session's active server whose title, artist, album, or composer match the search string parameter. In order to be included in the results list, a contiguous substring of any of these attributes must match the search string. The string comparison is case insensitive.

NOTE: Not all music servers support searching.

NOTE: The verbose-mode responses will only be sent when the RCP session's progress mode is set to verbose. See Transaction Progress in the Protocol Summary for more information. These progress responses were added in software version 2.6.19.

SetBrowseFilterArtist

Syntax:

SetBrowseFilterArtist filter-string

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no filter string given
OK – indicates success

Example:

SetBrowseFilterArtist Led Zeppelin
SetBrowseFilterArtist: OK

SetBrowseFilterArtist sets the RCP session's current artist browse filter to the filter-string given as a parameter. Browse filters may be used to limit the results of the browse listing commands. The artist filter will affect the results returned by ListSongs, ListAlbums, and on some servers, ListComposers and ListGenres.

Browse filters are case-sensitive and do not perform partial-matches. The filter term given for SetBrowseFilterArtist should be a result string from the ListArtists or SearchArtists commands.

NOTE: This browse filter is not supported by the Internet Radio server.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Until software version 2.5.155, the browse filters were global across all RCP sessions.

SetBrowseFilterAlbum

Syntax:

SetBrowseFilterAlbum filter-string

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no filter string given
OK – indicates success

Example:

SetBrowseFilterAlbum Presence
SetBrowseFilterAlbum: OK

SetBrowseFilterAlbum sets the RCP session's current album browse filter to the filter-string given as a parameter. Browse filters may be used to limit the results of the browse listing commands. The album filter will affect the results returned by ListSongs, ListArtists, and on some servers, ListComposers and ListGenres.

Browse filters are case-sensitive and do not perform partial-matches. The filter term given for SetBrowseFilterAlbum should be a result string from the ListAlbums or SearchAlbums commands.

NOTE: This browse filter is not supported by the Internet Radio server.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Until software version 2.5.155, the browse filters were global across all RCP sessions.

SetBrowseFilterComposer

Syntax:

SetBrowseFilterComposer filter-string

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no filter string given
OK – indicates success

Example:

SetBrowseFilterComposer Danny Elfman
SetBrowseFilterComposer: OK

SetBrowseFilterComposer sets the RCP session's current composer browse filter to the filter-string given as a parameter. Browse filters may be used to limit the results of the browse listing commands. The composer filter will affect the results returned by ListSongs, ListAlbums, ListArtists, and on some servers, ListGenres. Note that not all music servers support browsing by composer.

Browse filters are case-sensitive and do not perform partial-matches. The filter term given for SetBrowseFilterComposer should be a result string from the ListComposers or SearchComposers commands.

NOTE: This browse filter is not supported by the Internet Radio server.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Until software version 2.5.155, the browse filters were global across all RCP sessions.

SetBrowseFilterGenre

Syntax:

SetBrowseFilterGenre filter-string

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no filter string given
OK – indicates success

Example:

SetBrowseFilterGenre Rock
SetBrowseFilterGenre: OK

SetBrowseFilterGenre sets the RCP session's current genre browse filter to the filter-string given as a parameter. Browse filters may be used to limit the results of the browse listing commands. The genre filter will affect the results returned by ListSongs, ListAlbums, ListArtists, and on some servers, ListComposers.

Browse filters are case-sensitive and do not perform partial-matches. The filter term given for SetBrowseFilterGenre should be a result string from the ListGenres command.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Until software version 2.5.155, the browse filters were global across all RCP sessions.

SetBrowseFilterLocation

Syntax:

SetBrowseFilterLocation filter-string

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no filter string given
OK – indicates success

Example:

SetBrowseFilterLocation USA
SetBrowseFilterLocation: OK

SetBrowseFilterLocation sets the RCP session's current location browse filter to the filter-string given as a parameter. Browse filters may be used to limit the results of the browse listing commands. The location filter will only affect the results returned by ListSongs, and is only checked when connected to the Internet Radio server.

Browse filters are case-sensitive and do not perform partial-matches. The filter term given for SetBrowseFilterLocation should be a result string from the ListLocations command.

NOTE: The Internet Radio server does not currently allow multiple browse filters to be applied simultaneously, and it only checks the genre, location, and media language filters.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Browsing by location was added to RCP in 2.5.155.

SetBrowseFilterMediaLanguage

Syntax:

SetBrowseFilterMediaLanguage filter-string

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – no filter string given
OK – indicates success

Example:

SetBrowseFilterMediaLanguage French
SetBrowseFilterLocation: OK

SetBrowseFilterMediaLanguage sets the RCP session's current media language browse filter to the filter-string given as a parameter. Browse filters may be used to limit the results of the browse listing commands. The media language filter will only affect the results returned by ListSongs, and is only checked when connected to the Internet Radio server.

Browse filters are case-sensitive and do not perform partial-matches. The filter term given for SetBrowseFilterMediaLanguage should be a result string from the ListMediaLanguages command.

NOTE: The Internet Radio server, as of version 3.0.23, does allow multiple browse filters to be applied simultaneously, but it does not support certain filters, including artist, album and composer.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Browsing by media language was added to RCP in 2.5.155.

SetBrowseFilterTopStations

Syntax:

SetBrowseFilterTopStations

Classification:

Synchronous

Result Type:

No results

Responses:

OK – indicates success

Example:

SetBrowseFilterTopStations
SetBrowseFilterTopStations: OK

SetBrowseFilterTopStations sets the RCP session's current browse filter to return the top 25 Internet stations, ordered by the Radio Roku star rating. No argument is necessary, since this just sets a Boolean flag. Browse filters may be used to limit the results of the browse listing commands. The top stations filter will only affect the results returned by ListSongs, and is only checked when connected to the Internet Radio server.

When combined with other filter terms (*e.g.* genre or language), the command will return the 25 top stations matching the other criteria, ordered by rating. If there are not 25 stations matching the criteria specified, then the command will return fewer than 25 items (or possibly no items at all, if there were no matches).

For example, a client might use this command in combination with SetBrowseFilterGenre to return the top Rock stations.

NOTE: The Internet Radio server, as of version 3.0.23, does allow multiple browse filters to be applied simultaneously, but it does not support certain filters, including artist, album and composer.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Browsing by top station was added to RCP in 3.0.23.

SetBrowseFilterFavorites

Syntax:

SetBrowseFilterFavorites

Classification:

Synchronous

Result Type:

No results

Responses:

OK – indicates success

Example:

SetBrowseFilterFavorites
SetBrowseFilterFavorites: OK

SetBrowseFilterFavorites sets the RCP session's current browse filter to return only those Internet stations that the user has marked as "Favorite" stations. No argument is necessary, since this just sets a Boolean flag. Browse filters may be used to limit the results of the browse listing commands. The top stations filter will only affect the results returned by ListSongs, and is only checked when connected to the Internet Radio server.

Typically, this would be the only browse filter applied (to simply return the user's favorite stations). However, it can be combined with other terms to, for example, return the user's favorite Rock stations.

NOTE: Use of the Favorites feature in SoundBridge software 3.0.23 and later is dependent upon the unit's being associated with a Radio Roku user account. If the unit is not associated with a Radio Roku account, then any attempt to browse favorite stations will produce an empty list.

NOTE: The Internet Radio server, as of version 3.0.23, does allow multiple browse filters to be applied simultaneously, but it does not support certain filters, including artist, album and composer.

ISSUE: After setting browse filter, there is no way to clear it until calling one of the browse listing commands (like ListSongs, ListArtists, etc.).

HISTORICAL NOTE: Browsing by favorites was added to RCP in 3.0.23.

SetSongListSort

Syntax:

SetSongListSort alpha|albumTrack

Classification:

Synchronous

Result Type:

No results

Responses:

OK – indicates success
ParameterError – invalid option specified

Example:

SetSongListSort alpha
SetSongListSort: OK

SetSongListSort sets the RCP session's sorting option for any command that returns a list of songs. These commands include ListSongs, ListContainerContents, SearchSongs, and ListPlaylistSongs.

By default, the song-listing commands return songs in an undefined order (as determined by the server to which the RCP host is connected). Generally, ListPlaylistSongs will return the playlist's songs in playlist order, while the others will either be alphabetical or perhaps (as is common with iTunes), in the order in which tracks were added to the music collection.

The "alpha" option sorts the returned items alphabetically, and works with all of the commands listed above.

The "albumTrack" command sorts a list of songs first alphabetically by the name of the containing album, then by the disc number (if specified), then by the track number. However, not all song lists have enough information to perform this sort properly. In cases where the album and track information is not present, the tracks are sorted alphabetically.

A good rule of thumb with "albumTrack" is that if an artist name and/or album title has been specified with the SetBrowseFilterXXXX commands, then there will be enough information present to do the albumTrack sort. This information is also usually available for the contents of a playlist, but is not generally available for an unfiltered call to ListSongs, for example.

NOTE: It should be noted that, for implementation reasons, when browsing containers (using ListContainerContents), all container contents, including other containers, are treated as “songs” for sorting purposes.

HISTORICAL NOTE: SetSongListSort was added in SoundBridge software version 3.0.24.

SetBrowseListSort

Syntax:

SetBrowseListSort alpha|ignoreThe

Classification:

Synchronous

Result Type:

No results

Responses:

OK – indicates success
ParameterError – invalid option specified

Example:

SetBrowseListSort
SetBrowseListSort: OK

SetBrowseListSort sets the RCP session's sorting option for any command that returns a list of browse items. These commands include ListArtists, ListAlbums, ListComposers, ListGenres, ListLocations, and ListMediaLanguages, and the corresponding SearchXXXX counterparts.

By default, the browsing commands return items in an undefined order (as determined by the server to which the RCP host is connected). Most servers return these lists alphabetically, but this is not guaranteed.

The "alpha" option sorts the returned items alphabetically.

The "ignoreThe" command sorts the returned items alphabetically, but ignores the word "the" at the start of the item's text. So, "The Clash" will sort after "Carbon Leaf" but before "Coldplay". On the SoundBridge UI, this sort is only used when listing artists, but via RCP it is available for any of the commands noted above.

NOTE: It should be noted that, for implementation reasons, when browsing containers (using ListContainerContents), all container contents, including other containers, are treated as "songs" for sorting purposes, so SetSongListSort should be used in place of this command when navigating containers.

HISTORICAL NOTE: SetBrowseListSort was added in SoundBridge software version 3.0.24.

Getting Detailed Song Info

Overview

The following commands allow you to examine all attributes of the songs stored on a media server:

- GetSongInfo
- GetCurrentSongInfo

The RCP session's active server must be valid and connected for these commands to succeed. See the "Listing and Connecting to Media Servers" section for more details.

These commands are described in the pages that follow.

GetSongInfo

Syntax:

GetSongInfo n

Classification:

Transacted

Result Type:

Returns a song info object

Responses:

ParameterError – invalid song index

Example:

```
GetSongInfo 0
GetSongInfo: TransactionInitiated
GetSongInfo: id: 11453852
GetSongInfo: trackLengthMS: 384627
GetSongInfo: trackNumber: 9
GetSongInfo: format: MP3
GetSongInfo: status: unsupported
GetSongInfo: title: Lovely Day
GetSongInfo: artist: Bill Withers
GetSongInfo: album: Lean On Me
GetSongInfo: genre: Rock
GetSongInfo: comment: flac-to-mp3 version 1.5
GetSongInfo: songFormat: mp3
GetSongInfo: formatDescription: mp3 audio file
GetSongInfo: resource[0] url:
http://192.168.0.15/13852.mp3
GetSongInfo: resource[0] format: MP3
GetSongInfo: resource[0] bitrate: 128
GetSongInfo: resource[0] sampleRate: 44100
GetSongInfo: resource[0] sizeBytes: 6154036
GetSongInfo: OK
GetSongInfo: TransactionComplete
```

GetSongInfo reports all attributes of a song with a given index as reported by the currently connected server. The index argument refers to any list of songs generated by a browsing or searching command.

GetCurrentSongInfo

Syntax:

GetCurrentSongInfo

Classification:

Synchronous

Result Type:

Returns a song info object

Responses:

GenericError – no currently playing song

Example:

```
GetCurrentSongInfo
GetCurrentSongInfo: id: 11453938
GetCurrentSongInfo: trackLengthMS: 250292
GetCurrentSongInfo: year: 1999
GetCurrentSongInfo: trackNumber: 5
GetCurrentSongInfo: format: MP3
GetCurrentSongInfo: status: unsupported
GetCurrentSongInfo: title: Time 2 Get High
GetCurrentSongInfo: artist: Los Marijuanos
GetCurrentSongInfo: album: Puro Pleito
GetCurrentSongInfo: genre: Hip-Hop
GetCurrentSongInfo: composer: Pony Boy
GetCurrentSongInfo: songFormat: mp3
GetCurrentSongInfo: formatDescription: mp3 audio
     file
GetCurrentSongInfo: resource[0] url:
http://192.168.0.150:3689/db/11453938.mp3
GetCurrentSongInfo: resource[0] format: MP3
GetCurrentSongInfo: resource[0] bitrate: 128
GetCurrentSongInfo: resource[0] sampleRate: 44100
GetCurrentSongInfo: resource[0] sizeBytes: 405442
GetCurrentSongInfo: OK
```

GetCurrentSongInfo reports all attributes of the song currently playing as reported by the currently connected server.

Managing the Now Playing (ad-hoc) Playlist

The WMM maintains a list of songs that are currently scheduled for playback. This list is called the “Now Playing” playlist, or the “Ad-Hoc” playlist. The WMM offers several commands to the client for manipulating this list:

- NowPlayingClear
- ListNowPlayingQueue

These commands are described in the pages that follow.

NowPlayingClear

Syntax:

NowPlayingClear

Classification:

Synchronous

Result Type:

No result

Responses:

GenericError – no connected server

Example:

NowPlayingClear
NowPlayingClear: OK

NowPlayingClear clears the Now Playing playlist. This has the side effect that any media being played on the unit will be stopped and cleared from the queue.

ListNowPlayingQueue

Syntax:

ListNowPlayingQueue

Classification:

Synchronous

Result Type:

Returns a list of Song objects. This list can be used with the following commands:

- **GetSongInfo**
- **NowPlayingInsert**
- **PlayIndex**
- **QueueAndPlay**
- **QueueAndPlayOne**

Responses:

GenericError – no connected server

Example:

ListNowPlayingQueue
ListNowPlayingQueue: ListResultSize 3
ListNowPlayingQueue: Brick House
ListNowPlayingQueue: Insane in the Brain
ListNowPlayingQueue: Jamming
ListNowPlayingQueue: ListResultEnd

ListNowPlayingQueue lists all songs in the Now Playing playlist.

Initiating Media Playback

In order to simplify the user experience, the WMM provides commands that bundle queuing functionality with transport control in the same command. The following commands are examples:

- PlayIndex (jumps to a track indexed in the “Now-Playing” playlist and starts playback)
- NowPlayingInsert (inserts a single song from current item list into now playing queue, optionally at an offset into the queue)
- NowPlayingRemoveAt (removes a single song from the now playing queue at a given offset)
- QueueAndPlay (creates ad-hoc playlist from current item list and starts playback)
- QueueAndPlayOne (creates ad-hoc playlist from one item and plays)

These commands are described in the pages that follow.

PlayIndex

Syntax:

PlayIndex n

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError
OK – indicates success

Example:

PlayIndex 5
PlayIndex: OK

PlayIndex causes playback to begin at the beginning of the song in the "Now Playing" playlist at the track number n. This does not change the "Now Playing" playlist, so repeated issuances of the PlayIndex command with the same index argument will result in the same song being played (not the nth song after the currently playing song).

If there aren't any songs in the "Now Playing" queue, PlayIndex will result in a ParameterError.

NowPlayingInsert

Syntax:

NowPlayingInsert all|n [insert-index]

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError
GenericError
OK – indicates success

Example:

```
SearchSongs ok
SearchSongs: TransactionInitiated
SearchSongs: ListResultSize 2
SearchSongs: Boulevard Of Broken Dreams
SearchSongs: You Shook Me All Night Long
SearchSongs: ListResultEnd
SearchSongs: TransactionComplete
NowPlayingInsert 0
NowPlayingInsert: OK
ListNowPlayingQueue
ListNowPlayingQueue: ListResultSize 1
ListNowPlayingQueue: Boulevard Of Broken Dreams
ListNowPlayingQueue: ListResultEnd
```

NowPlayingInsert inserts a song into the Now Playing queue at the location specified by the optional insert-index parameter, based on the results of the most recent song browse or search (or of the current container). The previous contents of the Now Playing queue are shifted below the insertion point.

If the first parameter is the token “all”, then all songs in the current song result list will be added to the Now Playing queue.

NowPlayingInsert takes either one or two arguments. If a second argument is given, that argument is taken as the zero-based offset into the Now Playing queue at which to insert the song(s). If the second argument is not given, then the songs are appended to the Now Playing queue.

NowPlayingRemoveAt

Syntax:

NowPlayingRemoveAt n

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError
OK – indicates success

Example:

ListNowPlayingQueue
ListNowPlayingQueue: ListResultSize 2
ListNowPlayingQueue: Gold Dust Woman
ListNowPlayingQueue: Zealots
ListNowPlayingQueue: ListResultEnd
NowPlayingRemoveAt 0
NowPlayingRemoveAt: OK
ListNowPlayingQueue
ListNowPlayingQueue: ListResultSize 1
ListNowPlayingQueue: Zealots
ListNowPlayingQueue: ListResultEnd

NowPlayingRemoveAt removes a single song from the specified zero-based index of the “Now Playing” queue. The contents of the “Now Playing” queue following that index are shifted up by one.

QueueAndPlay

Syntax:

QueueAndPlay n

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError
OK – indicates success

Example:

```
SearchSongs ok
SearchSongs: TransactionInitiated
SearchSongs: ListResultSize 2
SearchSongs: Boulevard Of Broken Dreams
SearchSongs: You Shook Me All Night Long
SearchSongs: ListResultEnd
SearchSongs: TransactionComplete
QueueAndPlay 0
QueueAndPlay: OK
ListNowPlayingQueue
ListNowPlayingQueue: ListResultSize 2
ListNowPlayingQueue: Boulevard Of Broken Dreams
ListNowPlayingQueue: You Shook Me All Night Long
ListNowPlayingQueue: ListResultEnd
```

QueueAndPlay replaces the current Now Playing queue with the results of the most recent song browse or search (or of the current container), and then it begins playback of the list starting at the song at the zero-based index indicated by the parameter n. The previous contents of the Now Playing queue are deleted.

QueueAndPlayOne

Syntax:

QueueAndPlayOne n|working

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError
OK – indicates success

Example:

SearchSongs ok
SearchSongs: TransactionInitiated
SearchSongs: ListResultSize 2
SearchSongs: Boulevard Of Broken Dreams
SearchSongs: You Shook Me All Night Long
SearchSongs: ListResultEnd
SearchSongs: TransactionComplete
QueueAndPlayOne 0
QueueAndPlayOne: OK
ListNowPlayingQueue
ListNowPlayingQueue: ListResultSize 1
ListNowPlayingQueue: Boulevard Of Broken Dreams
ListNowPlayingQueue: ListResultEnd

QueueAndPlayOne replaces the current Now Playing queue with the song at the zero-based index given by parameter n from the most recent song browse or search (or of the current container), and begins playing that song. This causes the previous contents of the Now Playing queue to be deleted.

The parameter can also be the token string, “working”, which indicates that the contents of the RCP session’s current “working” song should be played. See the command SetWorkingSongInfo for more information on the “working” song.

Transport

Here are the transport commands for the WMM. These are the commands which start and stop playback, and which allow the client to move the “play head” around within the currently playing song or playlist.

- Play
- Pause
- Next
- Previous
- Stop
- Shuffle
- Repeat
- GetTransportState
- GetElapsedTime
- GetTotalTime
- GetCurrentNowPlayingIndex

These commands are described in the pages that follow.

Play

Syntax:

Play

Classification:

Synchronous

Result Type:

No result

Responses:

OK

Example:

Play
Play: OK

Play starts playback at the current track offset. If the RCP host is already playing, this command has no effect.

See also PlayPause

Pause

Syntax:

Pause

Classification:

Synchronous

Result Type:

No result

Responses:

OK

Example:

Pause
Pause: OK

Pause halts playback and maintains the position of playback in the current track. If the RCP host is not playing, this command has no effect.

See also PlayPause

PlayPause

Syntax:

PlayPause

Classification:

Synchronous

Result Type:

No result

Responses:

OK

Example:

PlayPause
PlayPause: OK

If the RCP host is playing, PlayPause halts playback and maintains the position of playback in the current track. If the RCP host is not playing, PlayPause starts playback at the current playback position.

See also Play, Pause

Next

Syntax:

Next

Classification:

Synchronous

Result Type:

No result

Responses:

OK

Example:

Next
Next: OK

Next moves the currently playing song to the next song in the Now Playing queue, and causes playback to begin (regardless of whether the unit was playing, paused, or stopped).

Previous

Syntax:

Previous

Classification:

Synchronous

Result Type:

No result

Responses:

OK

Example:

Previous
Previous: OK

Previous moves the currently playing song either to the beginning of the currently playing song, or to the Previous song in the Now Playing queue, and causes playback to begin (regardless of whether the unit was playing, paused, or stopped). If the song playback is between zero and five seconds after the start of the song, the Previous command causes playback to skip to the previous song. After five seconds, the Previous command causes playback to skip to the beginning of the current song.

Stop

Syntax:

Stop

Classification:

Synchronous

Result Type:

No result

Responses:

OK

Example:

Stop
Stop: OK

Stop stops the currently playing song (regardless of whether the unit was playing, paused, or stopped). If playback is resumed, it is always relative to the beginning of the first song in the Now Playing queue.

Shuffle

Syntax:

Shuffle [on|off|cycle]

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError
OK

Example:

Shuffle
Shuffle: off
Shuffle on
Shuffle: OK
Shuffle
Shuffle: on

With no arguments, Shuffle reports the state of the unit's shuffle-playback attribute. When this attribute is set to on, playback of the Now Playing queue (or any playlist) is performed in random order. When this attribute is set to off, playback is in order of the original queue or playlist. Given an argument of "on" or "off", Shuffle sets the attribute accordingly. Given an argument of "cycle", Shuffle toggles the attribute value.

Repeat

Syntax:

Repeat [one|all|none|cycle]

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError
OK

Example:

Repeat
Repeat: off
Repeat one
Repeat: OK
Repeat
Repeat: one

With no arguments, Repeat reports the state of the unit's repeat-playback attribute. When this attribute is set to one, playback of the current track is repeated after that track is done playing. When this attribute is set to all, playback continues to the end of the playlist then repeats from the beginning of the playlist. When this attribute is set to none, playback stops at the end of the playlist. Given an argument of "one", "all", or "none", Repeat sets the attribute accordingly. Given an argument of "cycle", Repeat cycles the value of the repeat attribute among each of the 3 options.

GetTransportState

Syntax:

GetTransportState

Classification:

Synchronous

Result Type:

No results

Responses:

GenericError
Play
Pause
Next
Prev
Stop
Buffering
Disconnected
Standby

Example:

GetTransportState
GetTransportState: Pause

GetTransportState reports the state of the unit's audio playback. The return values are Play, Pause, Next, Prev, Stop, Buffering, Disconnected, Standby, and Error.

GetElapsedTime

Syntax:

GetElapsedTime

Classification:

Synchronous

Result Type:

Returns a time string

Responses:

GenericError

Example:

GetElaspedTime
GetElaspedTime: 0:00:55

GetElapsedTime reports the hours, minutes, and seconds since the beginning of the currently playing song.

GetTotalTime

Syntax:

GetTotalTime

Classification:

Synchronous

Result Type:

Returns a time string

Responses:

GenericError

Example:

GetTotalTime
GetTotalTime: 0:03:15

GetElapsedTime reports the total duration of the currently playing song in hours, minutes, and seconds.

GetCurrentNowPlayingIndex

Syntax:

GetTransportState

Classification:

Synchronous

Result Type:

Returns a numeric result

Responses:

GenericError

Example:

GetCurrentNowPlayingIndex
GetCurrentNowPlayingIndex: 6

GetCurrentNowPlayingIndex reports the zero-based index into the Now Playing queue of the song that is being played back.

Volume Commands

These commands affect the unit playback volume:

- GetVolume
- SetVolume

These commands are described in the pages that follow.

GetVolume

Syntax:

GetVolume

Classification:

Synchronous

Result Type:

Returns a numeric result

Responses:

No responses

Example:

GetVolume
GetVolume: 57

GetVolume displays current volume setting of the RCP host. The volume setting is an integer between 0 and 100 (inclusive). The return value of GetVolume is always 0 when the unit is not connected to a media server, regardless of what the true stored volume setting may be.

SetVolume

Syntax:

SetVolume

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError
OK

Example:

SetVolume 65
SetVolume: OK

SetVolume sets the volume level used for playback. The integer argument n must be a number between 0 and 100, inclusive. The SetVolume command only works if the unit is connected to a media server.

Commands For Using Presets

User presets were introduced with software version 2.5, and provide a way for the user to quickly access favorite content at the press of a button. There are 18 user presets, organized as 3 banks of 6, and identified as A1 – A6, B1 – B6, and C1 – C6. The SoundBridge Radio has buttons on the top of the unit for accessing user presets, and all SoundBridge and WMM models can play presets in response to preset IR commands. (IR commands like "CK_PRESET_A1" are associated with user preset functionality, and can be dispatched from RCP using IrDispatchCommand.)

Any Now Playing playlist that can be initiated by hitting the PLAY button can be stored as a preset. E.g., a preset can represent "all tracks on the album Presence by Led Zeppelin." A preset can NOT represent Now Playing playlists that are created by using the ADD button to add songs from different browses or searches together.

When the unit is reset to factory defaults, the presets are set to a variety of internet radio stations.

Complete support for manipulating user presets is not yet available in RCP, but the following commands allow the RCP client access to a subset of their functionality. (Complete support will be available in a future software release.)

- ListPresets
- GetPresetInfo
- PlayPreset
- SetPreset
- GetWorkingSongInfo
- SetWorkingSongInfo
- ClearWorkingSong

These commands are described in the pages that follow.

HISTORICAL NOTE: These commands were added in software version 2.5.155.

Usage Scenario: Testing an Internet Radio URL

To play back an arbitrary Internet Radio URL from RCP, you must set the "working" song to identify the URL you want to play, make sure you are connected to an appropriate music server, and then execute the QueueAndPlayOne command.

First, we ensure that we're connected to the Internet Radio music server. Note that we set the server filter to "radio" before invoking the ListServers command, ensuring that the only list result is the built-in Internet Radio server, instead of

listing all servers and then trying to match the Internet Radio server by name or position. This is because the name of the Internet Radio server is localized and will vary depending on the RCP host's language settings.

```
GetConnectedServer
GetConnectedServer: OK
ServerDisconnect
ServerDisconnect: TransactionInitiated
ServerDisconnect: Disconnected
ServerDisconnect: TransactionComplete
SetServerFilter radio
SetServerFilter: OK
ListServers
ListServers: ListResultSize 1
ListServers: Internet Radio
ListServers: ListResultEnd
ServerConnect 0
ServerConnect: TransactionInitiated
ServerConnect: Connected
ServerConnect: TransactionComplete
```

Next, we playback our URL:

```
ClearWorkingSong
ClearWorkingSong: OK
SetWorkingSongInfo playlistURL
http://newsstream.publicradio.org
SetWorkingSongInfo: OK
SetWorkingSongInfo remoteStream 1
SetWorkingSongInfo: OK
QueueAndPlayOne working
QueueAndPlayOne: OK
```

If we wanted to save this internet radio stream as a user preset, we could then issue this command:

```
SetPreset A1 working
SetPreset: OK
```

Usage Scenario: Playing a Music File on the Local Network

The RCP client may wish to play back a music file stored on the local network that is not served by a music server. This is possible using the RCP session's working song variable. The RCP client must set up the working song's url and format fields and then call QueueAndPlayOne. It is recommended to not set the remoteStream field, as this will cause the file to be played back over automaticallyonce it reaches the end of the file.

The following commands will play back the music resource served by an http server on the local network at the URL http://172.17.21.10/cat.mp3. Note that the RCP host must be connected to a music server before the QueueAndPlayOne command is issued. (The Internet Radio server is a good candidate for this purpose.)

```
ClearWorkingSongInfo
ClearWorkingSongInfo: OK
SetWorkingSongInfo title Cat
SetWorkingSongInfo: OK
SetWorkingSongInfo url
http://172.17.21.10/cat.mp3
SetWorkingSongInfo: OK
SetWorkingSongInfo format MP3
SetWorkingSongInfo: OK
QueueAndPlayOne working
QueueAndPlayOne: OK
```

ListPresets

Syntax:

ListPresets

Classification:

Synchronous

Result Type:

Returns a list of preset titles

Responses:

No responses

Example:

```
ListPresets
ListPresets: ListResultSize 18
ListPresets: Minnesota Public Radio News
ListPresets: Container "Tracks"
ListPresets: Container "Sublime"
ListPresets: Container "Pearl Jam"
ListPresets: KQED
[13 records deleted]
ListPresets: ListResultEnd
```

ListPresets lists the titles of all user presets on the RCP host. The first 6 results correspond to presets A1 – A6, the next 6 results correspond to presets B1 – B6, and the final 6 results correspond to presets C1 – C6.

GetPresetInfo

Syntax:

GetPresetInfo n|preset-id

Classification:

Synchronous

Result Type:

Returns a user preset object

Responses:

ParameterError – invalid index or preset-id

Example:

```
GetPresetInfo A6
GetPresetInfo: preset: A6
GetPresetInfo: type: kInternetPreset
GetPresetInfo: name: KQED
GetPresetInfo: URL: http://www.kqed.org/kqed.asx
GetPresetInfo: filename:
GetPresetInfo: id:
GetPresetInfo: path:
GetPresetInfo: serverName: Internet Radio
GetPresetInfo: serverType: kFavoriteRadio
GetPresetInfo: frequency: 0
GetPresetInfo: filter genre:
GetPresetInfo: filter artist:
GetPresetInfo: filter composer:
GetPresetInfo: filter title:
GetPresetInfo: filter album:
GetPresetInfo: filter allFields:
GetPresetInfo: filter exactMatch: 0
GetPresetInfo: shuffleMode: off
GetPresetInfo: repeatMode: off
GetPresetInfo: OK
```

Gets detailed info about a user preset. The parameter identifying a user preset can either be a numeric zero-based index or a preset identifier of the form “A6”, where the first character is a capitalized letter ‘A’, ‘B’, or ‘C’, and the second character is a digit from 1 to 6.

PlayPreset

Syntax:

PlayPreset n|preset-id

Classification:

Synchronous

Result Type:

No results

Responses:

ErrorInitialSetupRequired – initial setup not complete
ErrorUpgradeInProgress – a software upgrade is in progress
ErrorAwaitingPostUpgradeReboot – a software upgrade has completed and unit needs reboot
ParameterError – invalid index or preset-id
ErrorNoServer – preset's server not available
ErrorNoNetwork – no network link detected
ErrorNotAllowedInDemoMode – preset access denied in demo mode
PowerStateOn – the unit came out of standby
OK – indicates success

Example:

PlayPreset A6
PlayPreset: OK

Plays back a user preset. The preset can be specified either by a numeric zero-based index or by a preset id token, of the form "A6" or "B3". This command is synchronous, and just queues a request of the playback mechanism to begin playback, which may begin up to several seconds later. The RCP client can monitor the progress of the playback with the GetTransportMode command, although there is no current way to get accurate failure notifications due to the automated nature of playback progress.

If the unit is in standby mode when this command is executed, the unit will come out of standby mode, and the RCP client will be notified with the following output:

PlayPreset: PowerStateOn

NOTE: ErrorInitialSetupRequired was added in software version 2.6.XXX.

SetPreset

Syntax:

GetPresetInfo n|preset-id working

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid index or preset-id

Example:

ClearWorkingSong
ClearWorkingSong: OK
SetWorkingSongInfo playlistURL http://coolradio.com/radio/
SetWorkingSongInfo: OK
SetWorkingSongInfo title Cool Radio
SetWorkingSongInfo: OK
SetPreset A6 working
SetPreset: OK

Sets a user preset. Current implementation sets an internet radio preset based on the working song. (See RCP command SetWorkingSongInfo.) The working song must have the internet radio station's URL in its "playlistURL" field. The preset name will be taken from the "title" field.

Support for setting other types of user presets will be available in a future software release.

GetWorkingSongInfo

Syntax:

GetWorkingSongInfo

Classification:

Synchronous

Result Type:

Returns a song info object

Responses:

No responses

Example:

```
GetWorkingSongInfo
GetWorkingSongInfo: id:
GetWorkingSongInfo: format: unknown
GetWorkingSongInfo: status: playable
GetWorkingSongInfo: title: Cool Radio
GetWorkingSongInfo: playlistURL:
http://coolradio.com/radio/
GetWorkingSongInfo: resource[0] url:
GetWorkingSongInfo: resource[0] format: unknown
GetWorkingSongInfo: OK
```

Gets detailed song info on the “working” song. Each RCP session has its own working song variable. The working song may be saved as an internet radio preset with the SetPreset command or queued for playback with the QueueAndPlayOne command.

SetWorkingSongInfo

Syntax:

SetWorkingSongInfo name value

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid field name
OK – indicates success

Example:

SetWorkingSongInfo title This Is The Title
SetWorkingSongInfo: OK

Sets one field in the "working" song from the given name-value pair parameter. Each RCP session has its own working song variable. The working song may be saved as an internet radio preset with the SetPreset command or queued for playback with the QueueAndPlayOne command. The field name must be followed by one space delimiter, and then the rest of the command line will be interpreted as the field data.

Valid field names:

```
id
trackLength
year
discNumber
discCount
trackNumber
trackCount
rating
bpm
startTimeMS
endTimeMS
volumeAdjust
tunerFrequency
compilation
disabled
remoteStream
format
status
title
artist
album
genre
composer
comment
songFormat
formatDescription
playlistURL
stationInfoURL
stationInfoString
location
language
url
format
bitrate
sampleRate
bitsPerSample
numChannels
sizeBytes
bigEndian
```

When setting the working song's format field, the following value names are accepted:

```
unknown
unsupported
MP3
AAC
AAC_DRM
WAV
AIFF
remotePLS
remoteM3U
remoteASX
remoteRhapsody
WMA
WMA_WMDRM
WMA_Rhapsody
WMA_Lossless
LPCM
container
playlist
AMRadio
FMRadio
microphone
```

ClearWorkingSong

Syntax:

ClearWorkingSong

Classification:

Synchronous

Result Type:

No results

Responses:

OK – indicates success

Example:

ClearWorkingSong
ClearWorkingSong: OK

Clears all fields in the RCP session's "working" song variable to a known empty state.

Socket Commands for Generalized Network Communication

RCP provides commands for performing basic network communication transactions through an interface somewhat akin to the Berkeley sockets model. These commands are intended for use by directly-connected clients of an embedded WMM, but they are available to any RCP client that wants to use them.

The intention of the RCP sockets commands is to allow arbitrary communications via TCP/IP streams and UDP datagrams, and to allow listening on a TCP port on the RCP host device and spawning new streams when outside clients connect to that port, but the current functionality is incomplete. Currently, RCP supports opening of TCP sockets to a remote host, and sending and receiving arbitrary amounts of data on those sockets.

The rest of the sockets functionality will be provided in a future software release.

The commands for generalized network communication are:

- SocketStreamOpen
- SocketClose
- SocketReceiveBytes
- SocketSendBytes
- SocketEventSubscribe
- SocketEventUnsubscribe

HISTORICAL NOTE: These 4 socket commands were added to RCP in software release 2.6.4. (As of this writing on 7/13/06, the 2.6 software is not available for general release.)

SocketStreamOpen

Syntax:

SocketStreamOpen ipaddr:port

Classification:

Transacted

Result Type:

Returns a numeric socket identifier that can be used with the following commands:

- **SocketReceiveBytes**
- **SocketSendBytes**
- **SocketClose**

Responses:

ErrorInitialSetupRequired – initial setup not complete
ParameterError
GenericError – indicates internal memory, resource, or other connection error
ResourceAllocationError – no sockets available
ErrorTimedOut – connect timed out
ErrorSocketConnectionRefused
ErrorSocketNotConnected – socket was shut down or reset by remote host
Connected – indicates success

Example:

SocketStreamOpen 172.17.2.253:4444
SocketStreamOpen: TransactionInitiated
SocketStreamOpen: SocketId: 15
SocketStreamOpen: Connected
SocketStreamOpen: TransactionComplete

This is a transacted command. It will return a socket id parameter which you must use in future socket commands to identify the socket.

The parameter must consist of a dotted IPv4 IP address (it can't be a name to be resolved via DNS) followed by a colon followed by a port number. All numbers are in standard base-10 (decimal) format.

Once opened, the RCP client is responsible for closing the socket with the SocketClose command. Failing to do so can result in the eventual socket starvation of the RCP host and will cause network failures in the normal operation of the RCP host as a network music player. Note that once SocketStreamOpen reports the Connected result, the socket will remain

allocated until the RCP client calls SocketClose, even if subsequent RCP commands (like SocketReceiveBytes, e.g.) return the ErrorSocketNotConnected result! This means that the socket has disconnected from the remote host, but it is still allocated on the RCP host.

The RCP host will not wait forever for a socket to connect to the remote host. The amount of time it waits before giving up is not guaranteed, but is currently about 70 seconds.

NOTE: The RCP host has a limited number of sockets, and these are shared among the competing resources of all RCP clients and the various networking subsystems employed by the SoundBridge / WMM architecture. RCP clients should use as an absolute maximum 16 sockets if they can guarantee that they are the only active RCP client, and should try to use a minimal number of sockets.

NOTE: Socket ids can get re-used after the original socket is closed.

NOTE: It is possible for a socket to connect without a transaction being initiated if the connection is with the localhost (127.0.0.1, e.g.)

NOTE: ErrorInitialSetupRequired was added in software version 2.6.XXX.

SocketClose

Syntax:

SocketClose socket-id

Classification:

Synchronous

Result Type:

No results

Responses:

ParameterError – invalid socket-id
OK – indicates success

Example:

SocketClose 16
SocketClose: OK

To close an open socket, use the SocketClose command. It takes one required parameter which is the socket id. Once a socket is closed, the RCP client can not use that socket id for any subsequent socket commands.

SocketReceiveBytes

Syntax:

SocketReceiveBytes socket-id [max-bytes]

Classification:

Synchronous

Result Type:

Returns binary data results

Responses:

ParameterError – invalid socket-id or max-bytes
GenericError – internal error
ErrorSocketNotConnected
ErrorSocketNoData

Example:

```
SocketReceiveBytes 16
SocketReceiveBytes: data bytes: 139
Welcome to the SoundBridge Shell version 2.5.145
release
Type '?' for help or 'help <command>' for help on
<command>.

SoundBridge>
```

To read from a connected socket, use the SocketReceiveBytes command. It takes one required parameter which is the socket id, and an additional optional paramter which is the maximum number of bytes to read from the socket, which is 1024 by default. The data is returned in the format specified by the command SetDataResultType, which defaults to a string of hex digits. To receive data in its raw binary form, make sure to execute the command, "SetDataResultType binary" before executing SocketReceiveBytes.

If the max-bytes parameter is omitted, then all the data currently received from the socket is returned.

In the above example, the data result type is set to binary, and we've connected this socket to the shell port (5555) of a SoundBridge device.

PLANNED FUNCTIONALITY: In a future software release, it is intended that if the max-bytes parameter is set to zero, then SocketReceiveBytes will return a result that indicates if any data is ready for receiving at that moment.

SocketSendBytes

Syntax:

SocketSendBytes socket-id num-bytes

Classification:

Synchronous (Interactive)

Result Type:

No results

Responses:

ParameterError – invalid socket-id or num-bytes
ResourceAllocationError – out of memory
ProtocolError – failure in data input handshake
ErrorSocketNotConnected
GenericError – send failure
OK – indicates success

Example:

SocketSendBytes 16 6
SocketSendBytes: DataInputReady 6
help<cr><lf>
SocketSendBytes: DataInputComplete
SocketSendBytes: OK

To send bytes to a connected socket, use the SocketSendBytes command. It takes two required parameters. First parameter is the socket id. Second parameter is the number of bytes to send. After executing the command, RCP will prompt you to send the data over the RCP connection.

IMPORTANT: SocketSendBytes initiates an interactive handshake process with the RCP client to upload the data to send from the RCP client to the RCP host. After the parameters are checked for validity, SocketSendBytes will output the "DataInputReady" token followed by a space and the number of bytes expected, followed by the line termination sequence. At this point, the RCP client must upload the binary data to send. Once the data has been received, SocketSendBytes will respond with the "DataInputComplete" token and initiate the network transaction.

The RCP session's data result type setting does NOT affect the way that the RCP client sends data to the RCP host during the data input handshaking procedure. The RCP client must always send binary data, and must send the exact number of bytes indicated as a parameter.

NOTE: Under consideration is the possibility of making SocketSendBytes become transacted when the local network buffers are not big enough to queue the data to send immediately. This does not happen now, though.

SocketEventSubscribe

SocketEventUnsubscribe

Syntax:

SocketEventSubscribe
SocketEventUnsubscribe

Classification:

Subscription

Result Type:

No results

Responses:

ErrorAlreadySubscribed – requested subscription but already subscribed
ErrorNotSubscribed – requested unsubscription but not currently subscribed
SocketReceiveReady – socket has data ready for reading
OK – indicates success

Example:

SocketEventSubscribe
SocketEventSubscribe: OK
SocketReceiveBytes 10
SocketReceiveBytes: ErrorSocketNoData
SocketEvent: SocketReceiveReady 10
SocketReceiveBytes 10
SocketReceiveBytes: data bytes: 7
HiThere

The RCP client may receive updates on status changes of its sockets asynchronously using the SocketEventSubscribe command. Once subscribed, a RCP client will receive event notifications for all sockets it has open. There is no way to selectively activate event notifications for a single socket only. Currently there is only one socket event, SocketReceiveReady.

The SocketReceiveReady event is sent when there is data ready to be read on a socket that the RCP client has previously opened. Once the event is sent, it will not be sent again until after the RCP client has executed the SocketReceiveBytes command successfully. The socket id is appended to the SocketReceiveReady notification with a space character delimiter.

UPnP Searching Commands

RCP exposes direct access to the UPnP discovery mechanism of the RCP host, for discovery of other UPnP devices on the network.

- UPnPSearch
- UPnPGetDeviceInfo
- UPnPGetDeviceDescription

HISTORICAL NOTE: UPnP Searching Commands were added in software version 2.6.3 for WMM, and are not yet available in general release for SoundBridge models.

UPnPSearch

Syntax:

UPnPSearch [search-term]

Classification:

Transacted

Result Type:

Returns dynamic search results while transacting; once complete, the search results can be used with the following commands:

- **UPnPGetDeviceInfo**
- **UPnPGetDeviceDescription**

Responses:

ErrorInitialSetupRequired – initial setup not complete
ErrorTransactionPending – device supports one active UPnP search from all RCP clients
GenericError – UPnP subsystem error

Example:

UPnPSearch urn:schemas-upnp-org:device:MediaServer:1
UPnPSearch: TransactionInitiated
UPnPSearch: 89665984-7466-0011-091a-a62f49107165
UPnPSearch: NLESWORMGIHUGDFKSSE
UPnPSearch: 120e6585-120a-4deb-ac35-e2aa6af0520d
UPnPSearch: ca2ff5ef-82b1-4d34-97f7-5a64bf01e2da
UPnPSearch: 08c8a64c-4e23-4a47-8905-45b0b483216b
UPnPSearch: TransactionComplete

UPnPSearch performs a search of the local network for UPnP devices. Using the search-term parameter, the RCP client can specify specific search criteria. If omitted, the search is performed for all UPnP devices. The search term must have one of the following formats:

- ssdp:all
- upnp:rootdevice
- uuid:<theUUID>
- urn:<theURN>

Searches for “ssdp:all” will search for all UPnP devices, and is equivalent to not passing a search term parameter. The search term “upnp:rootdevice” will search for all UPnP root devices. The search term “uuid:<theUUID>” will search for a specific UPnP device by UUID. (Replace <theUUID> with the actual UUID of the

device to search for.) The search term “urn:<theURN>” will search for devices that match the type specified by the urn term. See the example above for the expected format of this term.

NOTE: Only devices are currently searchable via the “urn:” specifier currently, although the UPnP spec allows searching for services as well.

IMPORTANT: While the UPnPSearch command is transacting, it behaves somewhat like a subscription command because it outputs search results as soon as they are received from the network. Importantly, these search results do NOT constitute list results, so they will not replace any existing RCP session list results. Also, it is important to note that UPnP search results are currently shared among all RCP sessions on an RCP host, so it is best to use the search results soon after generating them.

UPnPGetDeviceInfo

Syntax:

UPnPGetDeviceInfo n|uuid

Classification:

Synchronous

Result Type:

Returns a UPnP device info object

Responses:

ParameterError – invalid device specification

Example:

```
UPnPGetDeviceInfo 0
UPnPGetDeviceInfo: uuid: 89665984-7466-0011-091a-
a62f49107165
UPnPGetDeviceInfo: type: schemas-upnp-
org:device:MediaServer:1
UPnPGetDeviceInfo: deviceDescriptionURL:
http://172.17.0.60:9000/DeviceDescripti
on.xml
UPnPGetDeviceInfo: rootDevice: 1
UPnPGetDeviceInfo: service[0]: schemas-upnp-
org:service:ContentDirectory:1
UPnPGetDeviceInfo: service[1]: schemas-upnp-
org:service:ConnectionManager:1
UPnPGetDeviceInfo: OK
```

UPnPGetDeviceInfo returns the full information on a UPnP device that was gathered during a UPnP device search initiated with UPnPSearch. The RCP client must have already called UPnPSearch prior to executing this command. The parameter is either a zero-based index into the search results or the uuid of the device to get info for.

UPnPGetDeviceDescription

Syntax:

UPnPGetDeviceDescription n|uuid

Classification:

Transacted

Result Type:

Returns a UPnP device description object

Responses:

ParameterError – invalid device specification
ErrorTransactionPending – device supports one active device description lookup from all RCP clients
ErrorTimedOut – http request timed out
GenericError – UPnP subsystem error

Example:

```
UPnPGetDeviceDescription 0
UPnPGetDeviceDescription: TransactionInitiated
UPnPGetDeviceDescription: type: urn:schemas-upnp-org:device:MediaServer:1
UPnPGetDeviceDescription: friendlyName: [JOUST] MusicServer by TwonkyVision
UPnPGetDeviceDescription: manufacturer: TwonkyVision GmbH
UPnPGetDeviceDescription: manufacturerURL: http://www.twonkyvision.de
UPnPGetDeviceDescription: modelDescription: TwonkyVision Media Server
UPnPGetDeviceDescription: modelName: TwonkyVision Media Server
UPnPGetDeviceDescription: modelNumber: 1.0
UPnPGetDeviceDescription: modelURL: http://www.twonkyvision.de/UPnP
UPnPGetDeviceDescription: serialNumber:
UPnPGetDeviceDescription: udn: uuid:89665984-7466-0011-091a-a62f49107165
UPnPGetDeviceDescription: upc:
UPnPGetDeviceDescription: serviceType[0]: urn:schemas-upnp-
org:service:ContentDirectory:1
UPnPGetDeviceDescription: serviceId[0]: urn:upnp-org:serviceId:ContentDirectory
UPnPGetDeviceDescription: scpdURL[0]:
http://172.17.0.60:9000/ContentDirectory.xml
UPnPGetDeviceDescription: controlURL[0]:
http://172.17.0.60:9000/ContentDirectory/Control
UPnPGetDeviceDescription: eventSubURL[0]:
http://172.17.0.60:9000/ContentDirectory/Event
UPnPGetDeviceDescription: serviceType[1]: urn:schemas-upnp-
org:service:ConnectionManager:1
UPnPGetDeviceDescription: serviceId[1]: urn:upnp-
org:serviceId:ConnectionManager
UPnPGetDeviceDescription: scpdURL[1]:
http://172.17.0.60:9000/ConnectionManager.xml
UPnPGetDeviceDescription: controlURL[1]:
http://172.17.0.60:9000/ConnectionManager/Control
UPnPGetDeviceDescription: eventSubURL[1]:
http://172.17.0.60:9000/ConnectionManager/Event
UPnPGetDeviceDescription: TransactionComplete
```

UPnPGetDeviceDescription performs a network http request for the UPnP device description document and parses and returns the results for a UPnP device that was gathered during a UPnP device search initiated with UPnPSearch. The RCP client must have already called UPnPSearch prior to executing this command. The parameter is either a zero-based index into the search results or the uuid of the device to get info for.

Power State commands

RCP exposes commands to check and set the current power state of the RCP host.

- GetPowerState
- SetPowerState

HISTORICAL NOTE: These commands were added in 3.0.24.

GetPowerState

Syntax:

GetPowerState

Classification:

Synchronous

Result Type:

Returns a power state token

Responses:

No Responses

Example:

GetPowerState
GetPowerState: standby

GetPowerState returns the current power state of the system. Valid power states are:

- "standby" – the system is in standby
- "on" – the system is powered on (but not necessarily connected to a server)

SetPowerState

Syntax:

SetPowerState on|standby [yes|no]

Classification:

Synchronous

Result Type:

No result

Responses:

ParameterError – invalid power state or option
OK – indicates success

Example:

SetPowerState on no
SetPowerState: OK

SetPowerState sets the current power state of the system. Valid power states are:

- "standby" – the system is in standby, with no server connection
- "on" – the system is powered on (but not necessarily connected to a server)

The second parameter is *required*, but only for the "on" state. Pass "yes" to reconnect to the last-used server (if available), or "no" to power on but not connect to any server.

HISTORICAL NOTE: This command was added in version 3.0.24.