Russound I/O (RIO) Protocol for 3rd Party Integrators

Revision 1.06.00

Russound FMP
5 Forbes Road
Newmarket NH 03857

Document Information	
Dated:	October 31, 12
Submitted By:	Bill Edmondson

Document History

Note: Version identifiers are used throughout the document to indicate when a feature was introduced.

Revision	Date	Author	Version	Description
1.00.00	12/11/09	bille		Initial Release
1.01.00	7/6/10	bille	***v1.1***	Added support for Power Management
1.02.00	6/23/11	bille	***v1.2***	Added ability to GET most controller, zone, source and system parameters. Added DMS-3.1 support. Added clarification for case-sensitivity. Corrected case in S[s].type=”Home COntrol” to ”Home Control” Added clarification for using Media Management (one session per connection) Updated definitions of the S[s].channel and S[s].channelName keys Updated use of S[s].composerName
1.03.00	6/30/11	bille	***v1.3***	Updated DMS-3.1 MM support. Added KeyCode EVENT. Added GET S[s].ipAddress.
1.04.00	10/11/11	bille	***v1.4***	Added Shuffle support Added MMClose command Added zone page support

1.05.00	4/1/12	bille	***v1.5***	Added Repeat support (see the 'EVENT' section) Added Controller type support (see the 'GET' section) Added SET/GET System.language Support
1.06.00	7/15/12	bille	***v1.6***	Added System.language notification support Added Media Management Long List support Added Media Management Screen Templates Removed 'STANDBY' response from GET System.status command

Table Of Contents

Introduction **4**

Conventions **4**

Command Overview **4**

RIO Protocol Syntax **4**

RIO Command Response Syntax **4**

The ‘VERSION’ Command **4**

VERSION Examples **4**

The ‘GET’ Command **4**

GET Examples **4**

The ‘SET’ Command **4**

SET Examples **4**

The ‘ADJUST’ Command **4**

ADJUST Examples **4**

The ‘EVENT’ Command **4**

Physical vs Logical Source Selection **4**

Key Events **4**

EVENT Examples **4**

The ‘WATCH’ Command **4**

WATCH Expiration **4**

WATCH Notification Messages **5**

WATCH Examples **5**

Media Management **5**

RIO MM Commands/Responses **5**

RIO MM Commands **5**

Initialization **5**

Termination (v1.4) **5**

Menu Item Selection **5**

Screen Data Confirmation (v1.3) **5**

Item Navigation **5**

Screen Navigation **5**

Text Edit Navigation **5**

Text Field Entry **5**

RIO MM Responses **5**

Screen Change Notifications **5**

Text Field Change Notifications **5**

More on the ‘key’ string... **5**

Using PuTTY as a RIO Client **5**

Low Power Considerations **5**

RIO Keepalive **5**

Waking a RIO device using Wake On LAN **5**

Introduction

This document provides a description of a new protocol intended to ease integration of 3rd party devices and software with Russound C-Series and E-Series systems. We refer to this new feature as **Russound I/O**, or **RIO**.

RIO is a new text-based command set that improves upon our 3rd Party Integration Support, making it easier to develop UI devices and applications that require Russound system integration and control. Here is a summary of the services provided by this new command set:

- Serves as a superset of the existing RNet protocol capability
- Full 2-way communication capability
- The RIO Command Set is available as ASCII text via IP (using port 9621) and RS232 interfaces. Up to 8 simultaneous IP connections is supported for C-Series and E-Series systems.
- Adjustable RS232 baud rates (choose between 19200, 38400, 57600 and 115200) and protocol (choose between RNet and RIO), configured via SCS-C5
- Commands for realtime configuration and monitoring of system parameters such as Volume, Zone On/Off, Source Selection, Party Mode, DND, Source/Zone Names, Preamp Controls and more
- Support for asynchronous notification of activity on a per-zone, per-source and per-system basis.
- Support for Media Management (aka, 'Content Browsing and Selection'). This provides a compressed command set, making it easy to develop a UI device capable of mimicking the current Russound TS2 behavior. This presents (i.e. "pushes") the Menu choices and Media Lists from the C-Series and E-Series system to the 3rd party UI via a small set of RIO 'MM' events. With addition client development, it is possible to create a more robust interface by using a new set of ‘Long List’ functions, introduced in RIO v1.6.
- Support for submitting multiple commands in a single request (with multiple results in a single response)
-
- Support for low-power management (***v1.1***)
-
- Easy-to-understand error responses for illegal or malformed commands
- Full documentation of the RIO protocol

Conventions

This section describes conventions used throughout this document. Note that these document conventions are used for clarification only. The RIO Protocol is entirely case-insensitive. To be more specific, the case of incoming commands are ignored. Outgoing responses are formatted exactly as they appear in this document, regardless of the case of the incoming command.

RIO commands are specified in all uppercase. For example,

GET C[1].Z[4].currentSource

In this case, ‘GET’ is a RIO command.

Text enclosed in ‘< >’ provides a short description of variable text data that is part of the RIO message. For example,

S VERSION=“<version #>”

In this case, ‘version #’ is a short description that is replaced with the actual RIO version number at runtime.

“<key>” refers to the RIO command key string.

The key syntax is described in detail in the “More on the ‘key’ string...” section at the end of this document.

Command Overview

RIO defines a small set of commands, providing access to many of the capabilities of the Russound System. These commands are presented here in general terms, along with a description of the relevant parameters. All commands are available via serial and IP.

VERSION - request the version of the supported RIO protocol

GET - return the value of one or more system parameters

SET - modify the value of one or more system parameters

ADJUST - modify the relative value of one or more system parameters

EVENT - send an event from a zone

WATCH - register to receive asynchronous messages from a particular zone, source or system

RIO Protocol Syntax

This section presents each RIO Command and Response in detail.

The specifics of the RIO protocol are:

RIO commands are case-insensitive.

RIO commands are made up of ASCII characters except for the terminating characters.

All RIO commands must be terminated with a <CR> (0x0D hex)

All RIO responses are terminated with a <CR><LF> (0x0D 0x0A hex)

RIO Command Response Syntax

For RIO commands that are processed successfully, a response is sent with this format:

S <optional data>

For RIO commands that result in failure, a response is sent with this format:

E <error message>

For asyncronous RIO responses, or ‘notifications’, a response is sent with this format:

N <key>=”<value>”

The 'VERSION' Command

The *VERSION* command is used to request the version of the RIO protocol running on the RIO server device.

VERSION Command Syntax:

VERSION

VERSION Response:

S VERSION="<version #>"

VERSION Examples

1) Request the RIO Protocol version from an MCA-C5 controller

VERSION Command:

VERSION

VERSION Response:

S VERSION= "01.06.00"

The 'GET' Command

The *GET* command returns the value of one or more system parameters. This command is performed synchronously, returning the current value once upon request. A system parameter is addressed by a 'key' string. For more details on the 'key' string, see the section at the end of this document.

To request a single system parameter,

GET Command Syntax:

GET <key>

Successful Response:

S <key>="<value>"

To request multiple system parameters,

GET Command Syntax:

GET <key1>, <key2> ..., <keyN>

Successful Response:

S <key1>="<value1>", <key2>="<value2>", ..., <keyN>="<valueN>"

The tables below provide a complete list of the 'key' strings supported by the *GET* command.

Controller GET Key Table Controller table indices are identified by '*c*', a number from 1 to 6.		
Key	**Description**	**Range**
C[c].ipAddress (***v1.1***)	IP Address for the controller	a valid IP Address (xxx.xxx.xxx.xxx)
C[c].macAddress (***v1.1***)	MAC Address for the controller	a valid MAC Address (xx:xx:xx:xx:xx:xx)
C[c].type (***v1.5***)	Controller type (i.e., model)	MCA-C3 MCA-C5

System GET Key Table		
Key	**Description**	**Range**
System.status (***v1.2***)	System status	OFF/ON
System.language (***v1.5***)	System Language	ENGLISH/CHINESE/RUSSIAN

Zone GET Key Table Controller table indices are identified by '*c*', a number from 1 to 6. Zone table indices are identified by '*z*', a number from 1 to 8.		
Key	**Description**	**Range**
C[*c*].Z[*z*].name	Name of the specified zone	12 char max
C[*c*].Z[*z*].currentSource	Current physical Source selection for the zone	MCA-C5: 1 to 8
		ACA-E5: 1 to 12
C[*c*].Z[*z*].volume	Volume setting for the zone	0 to 50
C[*c*].Z[*z*].bass	Bass setting for the zone	-10 to 10
C[*c*].Z[*z*].treble	Treble setting for the zone	-10 to 10
C[*c*].Z[*z*].balance	Balance setting for the zone	-10 to 10
Notes:	Note: For the 'balance' parameter, a value of '-10' represents the leftmost position in the stereo spectrum, 0 represents the center and '10' represents the rightmost position.	
C[*c*].Z[*z*].loudness	Loudness setting for the zone	OFF/ON
C[*c*].Z[*z*].turnOnVolume	Turn On Volume setting for the zone	0 to 50
C[*c*].Z[*z*].doNotDisturb	Do Not Disturb setting for the zone	OFF/ON/SLAVE
C[*c*].Z[*z*].partyMode	Party Mode setting for the zone	OFF/ON/MASTER
C[c].Z[z].status	Power status for the zone	OFF/ON
C[c].Z[z].mute	Mute status for the zone	OFF/ON
C[c].Z[z].sharedSource	Shared source status for the zone	OFF/ON
C[c].Z[z].lastError	Error string triggered by the last zone operation	12 char max
C[c].Z[z].page (***v1.4***)	Page status for the zone	OFF/ON

Source GET Key Table Source table indices are identified by '*s*', a number from 1 to 12.		
Key	**Description**	**Range**
S[s].name	Name of the specified source	12 char max
S[s].type (***v1.2***)	Type of the specified source.	Amplifier Television Cable Video Acc Satellite VCR Laser Disc DVD Tuner / Amplifier Misc Audio CD Home Control 5 Disc CD Changer 6 Disc CD Changer CD Changer DVD Changer RNET AM/FM Tuner (Internal) RNET XM Tuner (Internal) RNET Sirius Tuner (Internal) RNET AM/FM Tuner (External) RNET XM Tuner (External) RNET Sirius Tuner (External) RNET SMS3 RNET iBridge Dock RNET iBridge Bay Arcam T32 DMS-3.1 Media Streamer (***v1.3***) DMS-3.1 AM/FM Tuner (***v1.3***)

Source GET Key Table Source table indices are identified by '*s*', a number from 1 to 12.			
Key		**Description**	**Range**
Notes:		A source that is not configured will return an empty string as it's type value. For example, if Source 4 has not been configured for use, sending: • GET S[4].type will produce this response: • S S[4].type=""	
S[s].composerName (***v1.2***)		Composer of the now playing content on the specified source.	37 char max
Notes:	The 'composerName' key is valid for these source types: • RNET Sirius Tuner (Internal) • RNET Sirius Tuner (External)		
S[s].ipAddress (***v1.3***)		IP Address for the specified source.	a valid IP Address (xxx.xxx.xxx.xxx)
Notes:	The 'ipAddress' key is valid for these source types: • RNET SMS3 • DMS-3.1 Media Streamer • DMS-3.1 AM/FM Tuner		
S[s].channel (***v1.2***)		Channel (frequency) of the now playing content on the specified source.	37 char max
Notes:	The 'channel' key is valid for these source types: • RNET AM/FM Tuner (Internal) • RNET AM/FM Tuner (External) • DMS-3.1 AM/FM Tuner (***v1.3***)		
S[s].coverArtURL (***v1.2***)		Cover Art URL for the now playing content on the specified source.	255 char max

Source GET Key Table Source table indices are identified by '*s*', a number from 1 to 12.			
Key		**Description**	**Range**
Notes:	The 'coverArtURL' key is valid for these source types: • DMS-3.1 Media Streamer (SiriusXM and Internet Radio only) (***v1.3***)		
S[s].channelName (***v1.2***)		Channel name of the now playing content on the specified source.	37 char max
Notes:	The 'channelName' key is valid for these source types: • RNET Sirius Tuner (Internal) • RNET Sirius Tuner (External) • RNET XM Tuner (Internal) • RNET XM Tuner (External) • Arcam T32 • DMS-3.1 Media Streamer (***v1.3***)		
S[s].genre (***v1.2***)		Genre of the now playing content on the specified source.	37 char max
Notes:	The 'genre' key is valid for these source types: • RNET Sirius Tuner (Internal) • RNET Sirius Tuner (External) • RNET XM Tuner (Internal) • RNET XM Tuner (External) • Arcam T32		
S[s].artistName (***v1.2***)		Artist of the now playing content on the specified source.	37 char max

Source GET Key Table Source table indices are identified by '*s*', a number from 1 to 12.			
Key		**Description**	**Range**
Notes:	The 'artistName' key is valid for these source types: • RNET Sirius Tuner (Internal) • RNET Sirius Tuner (External) • RNET XM Tuner (Internal) • RNET XM Tuner (External) • RNET iBridge Dock • RNET iBridge Bay • RNET SMS3 • DMS-3.1 Media Streamer (***v1.3***)		
S[s].albumName (***v1.2***)		Album for the now playing content on the specified source.	37 char max
Notes:	The 'albumName' key is valid for these source types: • RNET iBridge Dock • RNET iBridge Bay • RNET SMS3 • DMS-3.1 Media Streamer (***v1.3***)		
S[s].playlistName (***v1.2***)		Playlist for the now playing content on the specified source.	37 char max
Notes:	The 'playlistName' key is valid for these source types: • RNET iBridge Dock • RNET iBridge Bay • RNET SMS3 • DMS-3.1 Media Streamer (***v1.3***)		

Source GET Key Table Source table indices are identified by '*s*', a number from 1 to 12.			
Key		**Description**	**Range**
S[s].songName (***v1.2***)		Song for the now playing content on the specified source.	37 char max
Notes:	The 'songName' key is valid for these source types: • RNET Sirius Tuner (Internal) • RNET Sirius Tuner (External) • RNET XM Tuner (Internal) • RNET XM Tuner (External) • RNET iBridge Dock • RNET iBridge Bay • RNET SMS3 • DMS-3.1 Media Streamer (***v1.3***)		
S[s].programServiceName (***v1.2***)		Program Service Name (PSN) of the now playing content on the specified source.	37 char max
Notes:	The 'programServiceName' key is valid for these source types: • RNET AM/FM Tuner (Internal) • DMS-3.1 AM/FM Tuner (***v1.3***)		
S[s].radioText (***v1.2***)		First line of radio text for the now playing content on the specified source.	37 char max
Notes:	The 'radioText' key is valid for these source types: • RNET AM/FM Tuner (Internal) • DMS-3.1 AM/FM Tuner (***v1.3***) • Arcam T32		
S[s].radioText2 (***v1.2***) S[s].radioText3 S[s].radioText4		Second, third and fourth line of radio text for the now playing content on the specified source.	37 char max
Notes:	The 'radioText2/3/4' keys are valid for these source types: • Arcam T32		

Source GET Key Table Source table indices are identified by '*s*', a number from 1 to 12.			
Key		**Description**	**Range**
S[s].shuffleMode (***v1.2***)		Shuffle mode for the now playing content on the specified source.	OFF/SONG/ALBUM (iBridge) OFF/ON (DMS-3.1)
Notes:	The 'shuffleMode' key is valid for these source types: • RNET iBridge Dock • RNET iBridge Bay • DMS-3.1 Media Streamer (***v1.3***)		
S[s].repeatMode (***v1.5***)		Repeat mode for the now playing content on the specified source.	OFF/SINGLE/ALL
Notes:	The 'repeatMode' key is valid for these source types: • DMS-3.1 Media Streamer		
S[s].mode (***v1.2***)		Provider mode or streaming service for the now playing content on the specified source.	Unknown USB Media Server Pandora SiriusXM Internet Radio Rhapsody Last.fm AirPlay
Notes:	The 'mode' key is valid for these source types: • DMS-3.1 Media Streamer (***v1.3***)		
S[s].Support.MM.longList (***v1.6***)		Source supports MM Long List commands	TRUE/FALSE
Notes:	The 'Support.MM.longList' key is valid for these source types: • RNET iBridge Dock • RNET iBridge Bay • DMS-3.1 Media Streamer		

GET Examples

1) Get the value for the current source of controller 1, zone 4:

GET Command:

GET C[1].Z[4].currentSource

GET Response:

S C[1].Z[4].currentSource=”1”

2) Get the values for the bass and treble of controller 1, zone 4:

GET Command:

GET C[1].Z[4].bass, C[1].Z[4].treble

GET Response:

S C[1].Z[4].bass=”6”, C[1].Z[4].treble=”5”

3) Get the IP Address of controller 1:

GET Command:

GET C[1].ipAddress

GET Response:

S C[1].ipAddress=”192.168.1.10”

The ‘SET’ Command

The *SET* command changes one or more system parameters. A system parameter is addressed by a ‘key’ string. For more details on the ‘key’ string, see the section at the end of this document.

SET operations are not subject to the current state of the system. They may be utilized at any time, provided the controller/zone is present in the system.

To modify a single system parameter,

SET Command Syntax:

SET <key>=”<value>”

Successful Response:

S <key>=”<value>”

To modify multiple system parameters,

SET Command Syntax:

SET <key1>=”<value1>”, <key2>=”<value2>”, ..., <keyN>=”<valueN>”

Successful Response:

S <key1>=”<value1>”, <key2>=”<value2>”, ..., <keyN>=”<valueN>”

A successful response returns the modified value.

The table below provides a complete list of the 'key' strings supported by the *SET* command.

System SET Key Table		
Key	**Description**	**Range**
System.language (***v1.5***)	System Language	ENGLISH/CHINESE/RUSSIAN

Controller tables indices are identified by '*c*', a number from 1 to 6.
Zone tables indices are identified by '*z*', a number from 1 to 8.

Zone SET Key Table		
Key	**Description**	**Range**
C[*c*].Z[*z*].bass	Bass setting for the zone	-10 to 10
C[*c*].Z[*z*].treble	Treble setting for the zone	-10 to 10
C[*c*].Z[*z*].balance	Balance setting for the zone	-10 to 10
C[*c*].Z[*z*].loudness	Loudness setting for the zone	OFF/ON
C[*c*].Z[*z*].turnOnVolume	Turn On Volume setting for the zone	0 to 50

Note that for the 'balance' parameter, a value of '-10' represents the leftmost position in the stereo spectrum, 0 represents the center and '10' represents the rightmost position.

SET Examples

1) Set the value for the Turn On Volume of controller 1, zone 4:

SET Command:

SET C[1].Z[4].turnOnVolume=”25”

SET Response:

S C[1].Z[4].turnOnVolume=”25”

2) Set the values for the bass and treble of controller 1, zone 4:

SET Command:

SET C[1].Z[4].bass=”10”, C[1].Z[4].treble=”8”

SET Response:

S C[1].Z[4].bass=”10”, C[1].Z[4].treble=”8”

The ‘ADJUST’ Command

The *ADJUST* command increments or decrements the current value of one or more system parameters by one. A system parameter is addressed by a ‘key’ string. For more details on the ‘key’ string, see the section at the end of this document.

The ADJUST command is useful in implementing controls that are intended to make adjustments to a system parameter relative to their current value.

To modify a single system parameter,

ADJUST Command Syntax:

ADJUST <key>=”+1”
ADJUST <key>=”-1”

Successful Response:

S <key>=”<value>”

A successful response returns the modified value.

To modify multiple system parameters,

ADJUST Command Syntax:

ADJUST <key1>=”<value1>”, <key2>=”+/-<value2>”, ..., <keyN>=”+/-<value3>”

where <valueN> is +1 or -1

Successful Response:

S <key1>=”<value1>”, <key2>=”<value2>”, ..., <keyN>=”<valueN>”

where <valueN> is the adjusted value of the parameter

A successful response returns the modified value.

The table below provides a complete list of the 'key' strings supported by the *ADJUST* command. It also specifies the allowable adjustment range for each.

Adjustments that result in out-of-range data values will not result in an error. However, the value of the parameter, after executing the ADJUST command, are governed not to exceed the allowable parameter range. These allowable parameter ranges are specified as '**Data Range**' in the table below.

Controller tables indices are identified by '*c*', a number from 1 to 6.
Zone tables indices are identified by '*z*', a number from 1 to 8.

Key	Description	Data Range
C[*c*].Z[*z*].bass	Bass setting for the zone	-10 to +10
C[*c*].Z[*z*].treble	Treble setting for the zone	-10 to +10
C[*c*].Z[*z*].balance	Balance setting for the zone	-10 to +10
C[*c*].Z[*z*].turnOnVolume	Turn On Volume setting for the zone	0 to 50

Note that for the 'balance' parameter, a value of '-10' represents the leftmost position in the stereo spectrum, 0 represents the center and '10' represents the rightmost position.

ADJUST Examples

1) Increase the value for the ***turn on volume*** of controller 1, zone 4, currently set to a value of ’20’:

ADJUST Command:

ADJUST C[1].Z[4].turnOnVolume=”+1”

ADJUST Response:

S C[1].Z[4].turnOnVolume=”21”

2) For controller 1, zone 1, simultaneously increase the value of the ***bass,*** currently set to a value of ‘1’ and lower the value of the ***treble,*** currently set to a value of ‘-2’:

ADJUST Command:

ADJUST C[1].Z[4].bass=”+1”, C[1].Z[4].treble=”-1”

ADJUST Response:

S C[1].Z[4].bass=”2”, C[1].Z[4].treble=”-3”

The 'EVENT' Command

The *EVENT* command is typically used to issue commands that are triggered by user actions (i.e., button presses, screen selections, etc). These commands may change system parameter values (such as zone volume adjustments). Unlike the SET and ADJUST commands, the EVENT commands may also affect system state (such as zone on/off status, party mode state, ...), depending on current conditions.

Due to their stateful behavior, executing the same EVENT command can provide different resultant values. For example, setting a controller/zone Party Mode to 'ON', when no Party is ongoing, will result in setting that controller/zone to Party Mode 'MASTER' status, since Party Mode requires at least one Master controller/zone.

Events are directed at a controller/zone pair and specified by an *Event ID and one or two event-specific data values.*

EVENT Command Syntax:

EVENT C[c].Z[z]!<event id> <data1> <data2>

Successful Response:

S

Physical vs Logical Source Selection

Source selection can be performed in two ways; physically and logically.

A physical source selection treats the supplied source number in terms of the source inputs as they appear on the rear panel of the System Controller.

Physical source selection is accessed using this syntax:

EVENT C[c].Z[z]!SelectSource <physical source number>

A logical source selection ignores the 'excluded' and 'unconfigured' sources. That is, the available sources (on a per-zone basis) are numbered from 1 to N, where *N* is the total number of available sources. Note that the Russound System Remote Control refers to the sources as 'logical' sources.

Logical source selection is accessed using this syntax:

EVENT C[c].Z[z]!KeyRelease SelectSource <logical source number>

The table below provides a complete list of the id's supported by the EVENT command.

Description	Event ID	Data 1 Range	Data 2 Range
Select a physical source	SelectSource	MCA-C5: 1 to 8 ACA-E5: 1 to 12	N/A
Turn a zone on	ZoneOn	N/A	N/A
Turn a zone off	ZoneOff	N/A	N/A
Turn all zones on	AllOn	N/A	N/A
Turn all zones off	AllOff	N/A	N/A
Send a Key Press	KeyPress	<key code>	N/A (except Volume)
Send a Key Release	KeyRelease	<key code>	N/A (except Source Select)
Send a Key Hold	KeyHold	<key code>	hold time (in msec)
Send a UEI Key Code (***v1.3***)	KeyCode	<1-100>	N/A
Change Party Mode	PartyMode	off/on/master	N/A
Change DND	DoNotDisturb	off/on	N/A
Change Shuffle Mode (***v1.4***)	Shuffle	N/A	N/A
Change Repeat Mode (***v1.5***)	Repeat	N/A	N/A

Key Events

Key Events are intended to emulate all of the buttons available on the Russound System Remote Control. They provide the Press/Release/Hold conditions, where applicable. The following tables list the EVENT Key Codes for each of these conditions.

This table lists the key codes that are supported by the **KeyPress** EVENT.

RIO Key Codes
Volume (0 to 50)
VolumeUp
VolumeDown

This table lists the key codes that are supported by the **KeyRelease** EVENT. Entries identified with a '*' are configurable via the SCS command editor (applicable to non-RNET sources only).

RIO Key Codes	
DigitZero	Enter
DigitOne	Last
DigitTwo	Sleep
DigitThree	Guide
DigitFour	Exit
DigitFive	MenuLeft
DigitSix	MenuRight
DigitSeven	MenuUp
DigitEight	MenuDown
DigitNine	Select
*Previous	Info
*Next	Menu
*ChannelUp	Record
*ChannelDown	PageUp
NextSource	PageDown
Power	Disc
*Stop	Mute
*Pause	
Favorite1	
Favorite2	
*Play	
SelectSource (1 to 12, logical)	

This table lists the key codes that are supported by the **KeyHold** EVENT.

KeyHold events must be accompanied by a 'hold time' parameter. The hold time is specified in milliseconds.

In order for the KeyHold EVENTs to operate correctly, they must be executed in a specific manner by the 3rd party device (the RIO 'client'). The KeyHold EVENT message must be transmitted once every 150 milliseconds for as long as the button is held. The 'hold time' parameter should be increased by 150 each time it is retransmitted. When the button is released, a KeyRelease EVENT command must be sent to complete the hold operation. See EVENT Example #2 for more details.

Entries identified with a '*' are configurable via the SCS command editor (applicable to non-RNET sources only).

RIO Key Codes		
DigitZero	*Stop	Select
DigitOne	*Pause	Info
DigitTwo	Favorite1	Menu
DigitThree	Favorite2	Record
DigitFour	*Play	PageUp
DigitFive	Mute	PageDown
DigitSix	Enter	Disc
DigitSeven	Last	
DigitEight	Sleep	
DigitNine	Guide	
*Previous	Exit	
*Next	MenuLeft	
*ChannelUp	MenuRight	
*ChannelDown	MenuUp	
Power	MenuDown	

EVENT Examples

1) Increase the volume of controller 1, zone 4:

EVENT Command:

EVENT C[1].Z[4]!KeyPress VolumeUp

EVENT Response:

S

2) Perform a Search Forward for approximately 1 second on an iPod for controller 1, zone 4:

EVENT Commands:

<user presses key>
<150msec time delay>
EVENT C[1].Z[4]!KeyHold Next 150
<150msec time delay>
EVENT C[1].Z[4]!KeyHold Next 300
<150msec time delay>
EVENT C[1].Z[4]!KeyHold Next 450
<150msec time delay>
EVENT C[1].Z[4]!KeyHold Next 600
<150msec time delay>
EVENT C[1].Z[4]!KeyHold Next 750
<150msec time delay>
EVENT C[1].Z[4]!KeyHold Next 900
<150msec time delay>
EVENT C[1].Z[4]!KeyHold Next 1050
<user releases key>
EVENT C[1].Z[4]!KeyRelease Next

EVENT Response:

S *<for each EVENT received>*

The 'WATCH' Command

The *WATCH* command enables a device to register for and receive asynchronous notifications of system parameter changes. The *WATCH* command groups the system parameters into categories: ZONE, SOURCE and SYSTEM.

Enabling a WATCH on a zone is particularly powerful. In addition to receiving notifications on zone parameters (such as status, volume, etc), the RIO client will also receive notifications for changes to the current source and it's parameters (such as songName, shuffleMode, etc).

The WATCH command is an excellent way for a UI device to remain aware of system status, allow the device to display current information with minimal communication or overhead. Each notification message is uniquely identified with a key string. This allows the UI device a way to identify relevant data, while filtering unneeded data. These UI device decisions are sometimes made on a screen-by-screen basis, or a stateful manner.

When the command is issued with the parameter set to 'ON', it will provide a snapshot of the system parameters in the requested category. Subsequent changes will be sent to the requesting device as their values change. These asynchronous change notifications will continue until the WATCH command is turned 'OFF' by the user or when the WATCH command expires (when the EXPIRESIN option is specified).

The WATCH command allows the user to specify a WATCH 'duration'. This value denotes the number of minutes that a particular WATCH operation will remain in effect. The RIO Server will indicate when a WATCH is about to expire with an 'EXPIRING' notification. It will also indicate the expiration condition with an 'EXPIRED' notification.

WATCH Command Syntax:

Start a Zone WATCH session

WATCH C[c].Z[z] ON

Start a Source WATCH session

WATCH S[s] ON

Start a System WATCH session

WATCH System ON

Successful Response:

S

N <key1>=”<value1>”
N <key2>=”<value2>”
...
N <keyN>=”<valueN>”

WATCH Command Syntax (continued):

Start a Zone WATCH session with an expiration time

WATCH C[c].Z[z] ON EXPIRESIN <time in minutes>

Start a Source WATCH session with an expiration time

WATCH S[s] ON EXPIRESIN <time in minutes>

Start a System WATCH session with an expiration time

WATCH System ON EXPIRESIN <time in minutes>

Successful Response:

S

N <key1>=”<value1>”
N <key2>=”<value2>”
...
N <keyN>=”<valueN>”

Stop a Zone WATCH session

WATCH C[c].Z[z] OFF

Stop a Source WATCH session

WATCH S[s] OFF

Stop a System WATCH session

WATCH System OFF

Successful Response:

S

WATCH Expiration

The WATCH command allows the 3rd party device to expire it's 'ON' condition after a specified number of minutes. The number of minutes to expire the WATCH command for a particular category (and it's instance) is specified using the optional 'EXPIRESIN' argument.

The 3rd party will receive a notification within 1 minute of impending expiration and another notification upon expiration.

The notifications are:

N EXPIRING=<watch type>
N EXPIRED=<watch type>

WATCH Notification Messages

The type of notification messages that result from enabling WATCH vary based on the WATCH parameter and current source type. The following sections list the notifications in these various scenarios.

WATCH System

Notifications:

N System.status="<value>"

For status, possible values are OFF, ON and STANDBY. (***v1.1***)

N System.language="<value>" (***v1.6***)

For language, possible values are ENGLISH, CHINESE and RUSSIAN.

WATCH a Zone

Notifications:

N C[c].Z[z].name=”<value>”
N C[c].Z[z].status=”<value>”
N C[c].Z[z].currentSource=”<value>”
N C[c].Z[z].volume=”<value>”
N C[c].Z[z].bass=”<value>”
N C[c].Z[z].treble=”<value>”
N C[c].Z[z].balance=”<value>”
N C[c].Z[z].loudness=”<value>”
N C[c].Z[z].doNotDisturb=”<value>”
N C[c].Z[z].partyMode=”<value>”
N C[c].Z[z].turnOnVolume=”<value>”
N C[c].Z[z].mute=”<value>”
N C[c].Z[z].sharedSource=”<value>”
N C[c].Z[z].lastError=”<value>”

N C[c].Z[z].page=”<value>” (***v1.4***)

WATCH a Source

For all source types

Notifications:

N S[s].type=”<value>”

where value is one of these possible source types

• Amplifier
• Television
• Cable
• Video Acc
• Satellite
• VCR
• Laser Disc
• DVD
• Tuner / Amplifier
• Misc Audio
• CD
• Home Control
• 5 Disc CD Changer
• 6 Disc CD Changer
• CD Changer
• DVD Changer
• RNET AM/FM Tuner (Internal)
• RNET XM Tuner (Internal)
• RNET Sirius Tuner (Internal)
• RNET AM/FM Tuner (External)
• RNET XM Tuner (External)
• RNET Sirius Tuner (External)
• RNET SMS3
• RNET iBridge Dock
• RNET iBridge Bay
• Arcam T32
• DMS-3.1 Media Streamer (***v1.3***)
• DMS-3.1 AM/FM Tuner (***v1.3***)

N S[s].name=”<value>”

WATCH a Source (continued)

For Sirius Satellite Radio Sources

N S[s].composerName=”<value>”
N S[s].channelName=”<value>”
N S[s].genre=”<value>”
N S[s].artistName=”<value>”
N S[s].songName=”<value>”

For XM Satellite Radio Sources

N S[s].channelName=”<value>”
N S[s].genre=”<value>”
N S[s].artistName=”<value>”
N S[s].songName=”<value>”

For AM/FM Tuner Sources

N S[s].channel=”<value>”

For AM/FM Tuner Sources with RDS

N S[s].programServiceName=”<value>”
N S[s].radioText=”<value>”
N S[s].channel=”<value>”

For DAB Tuner Source

N S[s].channelName=”<value>”
N S[s].genre=”<value>”
N S[s].radioText=”<value>”
N S[s].radioText2=”<value>”
N S[s].radioText3=”<value>”
N S[s].radioText4=”<value>”

WATCH a Source (continued)

For iBridgeDock and iBridgeBay Media Players

N S[s].artistName="<value>"
N S[s].albumName="<value>"
N S[s].playlistName="<value>"
N S[s].songName="<value>"
N S[s].shuffleMode="<value>"

For shuffleMode, possible values are 'OFF', 'SONG' and 'ALBUM'.

For SMS3 Media Players

N S[s].artistName="<value>"
N S[s].albumName="<value>"
N S[s].playlistName="<value>"
N S[s].songName="<value>"

For DMS-3.1 Media Streamers (***v1.3***)

N S[s].artistName="<value>"
N S[s].albumName="<value>"
N S[s].playlistName="<value>"
N S[s].songName="<value>"
N S[s].mode="<value>"
N S[s].channelName="<value>"
N S[s].coverArtURL="<value>"

N S[s].shuffleMode="<value>" (***v1.4***)

For shuffleMode, possible values are 'OFF' and 'ON'.

N S[2].repeatMode="<value>"(***v1.5***)

For repeatMode, possible values are 'OFF', 'SINGLE' and 'ON'.

WATCH Examples

1) WATCH for asynchronous changes on Controller 1, Zone 4:

WATCH Command:

WATCH C[1].Z[4] ON

WATCH Response:

S

N C[1].Z[4].status=”ON”
N C[1].Z[4].volume=”20”
N C[1].Z[4].bass=”10”
N C[1].Z[4].treble=”10”
N C[1].Z[4].balance=”10”
N C[1].Z[4].loudness=”OFF”
N C[1].Z[4].currentSource=”2”
N S[2].artist=”The Beatles”
N S[2].album=”Abbey Road”
N S[2].song=”Come Together”

WATCH related notifications:

A song begins to play on the Zone 4 current source...

N S[2].artist=”ABBA”
N S[2].album=”Arrival”
N S[2].song=”Dancing Queen”

Volume is adjusted on Zone 4...

N C[1].Z[4].volume=”21”

Media Management

RIO includes a set of commands specifically for Media Management. This allows for content filtering, browsing and selection. These commands provide access to media contained on all RNET Media Player source devices.

RIO supports a Media Management model that allows a simple implementation to mimic the existing navigation model used on the popular Russound UNO-TS2 touchscreen keypad. By implementing about 20 screen templates, each containing a small set of fixed buttons and static text fields, it is possible to provide full Media Management of all RNET Media Players.

These screen templates are presented in the Russound MediaManagement Protocol PDF document and can be found along with this document in the Third Party Development Toolkit. For more information on these text fields messages, see the ***Screen Change Notifications*** section.

RIO allows for one Media Management session per connection. An RS232 interface is treated as a single connection. Each IP socket is treated as a single connection.

Russound Media Management is only supported by RNET Source Devices that are considered ‘Media Players’. As of this writing, this includes:

• iBridge Bay
• iBridge Dock
• SMS3
• DMS-3.1 Media Streamer (***v1.3***)

RIO MM Commands/Responses

This section presents the RIO commands and responses used to access Media Management functionality.

RIO MM Commands

Initialization

This event restarts MM state within the source device. The menu navigation is reset to the topmost (i.e., home) menu.

EVENT C[c].Z[z]!MMInit

Termination (*v1.4*)

This event ends the MM session and returns the RIO session to the NowPlaying screen.

EVENT C[c].Z[z]!MMClose

After issuing this RIO MM command, a source-specific NowPlaying screen notification will be sent to the RIO client.

Menu Item Selection

This EVENT is used to select from a list of items presented by the previous set of text field notifications (see **Text Field Change Notifications** below). There are up to six valid selections, depending on the current screen being displayed.

EVENT C[c].Z[z]!MMSelectItem [1-6]

Screen Data Confirmation (*v1.3*)

This EVENT is used as confirmation for the current screen data and advances the Media Management session to the next navigable screen. Currently, the DMS-3.1 Info and TextEntry screens make use of this MM EVENT.

EVENT C[c].Z[z]!MMSelectOk

Item Navigation

These EVENTs are used to request the next and previous set of items from the current list.

EVENT C[c].Z[z]!MMNextItems

EVENT C[c].Z[z]!MMPrevItems

Screen Navigation

This EVENT requests that the previous screen be selected. This may result in the system sending a screen change notification (see **Screen Change Notifications** below) as well as text field change notifications (see **Text Field Change Notifications** below).

EVENT C[c].Z[z]!MMPrevScreen

Text Edit Navigation

These EVENTs allow editing of alphanumeric fields, where the text search feature is available.

EVENT C[c].Z[z]!MMCursorNext

EVENT C[c].Z[z]!MMCursorPrev

EVENT C[c].Z[z]!MMLetterUp

EVENT C[c].Z[z]!MMLetterDown

Text Field Entry

This EVENT allow direct entry of a text field, where the text search feature is available. This approach should be used by advanced UI devices capable of presenting an ASCII keyboard to the user for text entry. Currently, the DMS-3.1 TextEntry screens make use of this MM EVENT.

EVENT C[c].Z[z]!MMTextField "<text string>"(***v1.3***)

RIO MM Responses

Screen Change Notifications

This section presents the RIO messages ***received by*** the 3rd party device to indicate that a new screen template must be displayed. This RIO message is typically followed by a series of RIO messages containing text, used to populate the text fields within the screen to be displayed. For more information on these text fields messages, see the ***Text Field Change Notifications*** section.

These screen templates are presented in the Russound MediaManagement Protocol PDF document and can be found along with this document in the Third Party Development Toolkit. Each Screen Template is referenced by it's name and designator (1a thru 1u) in the Screen Change Notification section later in this document.

N S[s].MMScreen="<Screen ID>"

where Screen ID is defined as:

IBridge Bay/iBridge Dock Screen Identifiers

Screen ID	Screen Name
iPodRequestScreen	iPod Request
iPodPlaylistsScreen	iPod Request by Playlist
iPodGenresScreen	iPod Request by Genre
iPodArtistsScreen	iPod Request by Artist
iPodAlbumsScreen	iPod Request by Album
iPodSongsScreen	iPod Request by Song Title
iPodPlaylistOptionsScreen	iPod Playlist Options
iPodGenreOptionsScreen	iPod Genre Options
iPodArtistOptionsScreen	iPod Artist Options
iPodAlbumOptionsScreen	iPod Album Options

SMS3 Screen Identifiers

Screen ID	Screen Name
SMS3RequestScreen	SMS3 Request
SMS3ThemesScreen	SMS3 Request by Theme
SMS3GenresScreen	SMS3 Request by Genre
SMS3ArtistsScreen	SMS3 Request by Artist
SMS3AlbumsScreen	SMS3 Request by Album
SMS3SongsScreen	SMS3 Request by Song Title
SMS3InternetRadioScreen	SMS3 Request by Internet Radio
SMS3PlayOptionsScreen	SMS3 Play Artist
SMS3AlbumsByArtistScreen	SMS3 Play Album by Artist

DMS-3.1 Screen Identifiers (***v1.3***)

Screen ID	Screen Name
SourceMenuScreen	Menu
SourceInfoScreen	Info

Screen ID	Screen Name
SourceNowPlayingScreen	Now Playing
SourceTextEntryScreen	Text Entry

Text Field Change Notifications

This section presents the RIO notification messages that are sent by the Russound RIO system.

Menu screens can expect notifications of changes to Menu Item text. These will appear as notification messages in this format:

N S[s].MMMenuItem[1-6].text=”<text string>”

All DMS-3.1 screens, listed above, can expect notifications of changes to the screen title. These will appear as notification messages in this format:

N S[s].MMTitle.text=”<title string>” (***v1.3***)

The DMS-3.1 Info screen, listed above, can expect notifications of changes to the info fields. These will appear as notification messages in this format:

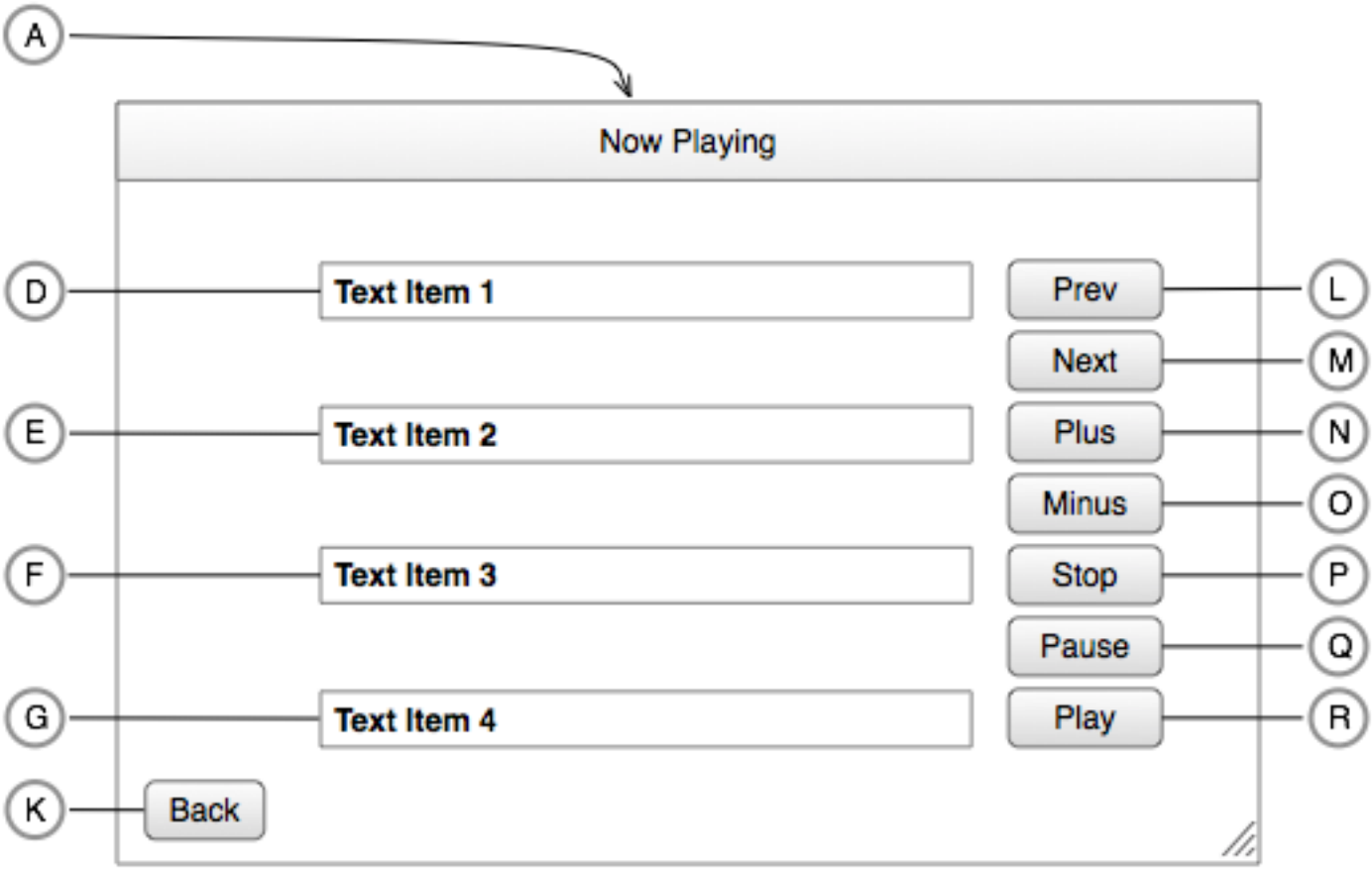
N S[s].MMInfo[1-4].text=”<info string>” (***v1.3***)

The DMS-3.1 Text Entry screen, listed above, can expect notifications of changes to the text entry field (where user text is entered) and help text (describing what type of information the user should enter into the text entry field. These will appear as notification messages in this format:

N S[s].MMTextField.text=”<text string>” (***v1.3***)

N S[s].MMHelp.text=”<text string>”

iPod Now Playing Screen

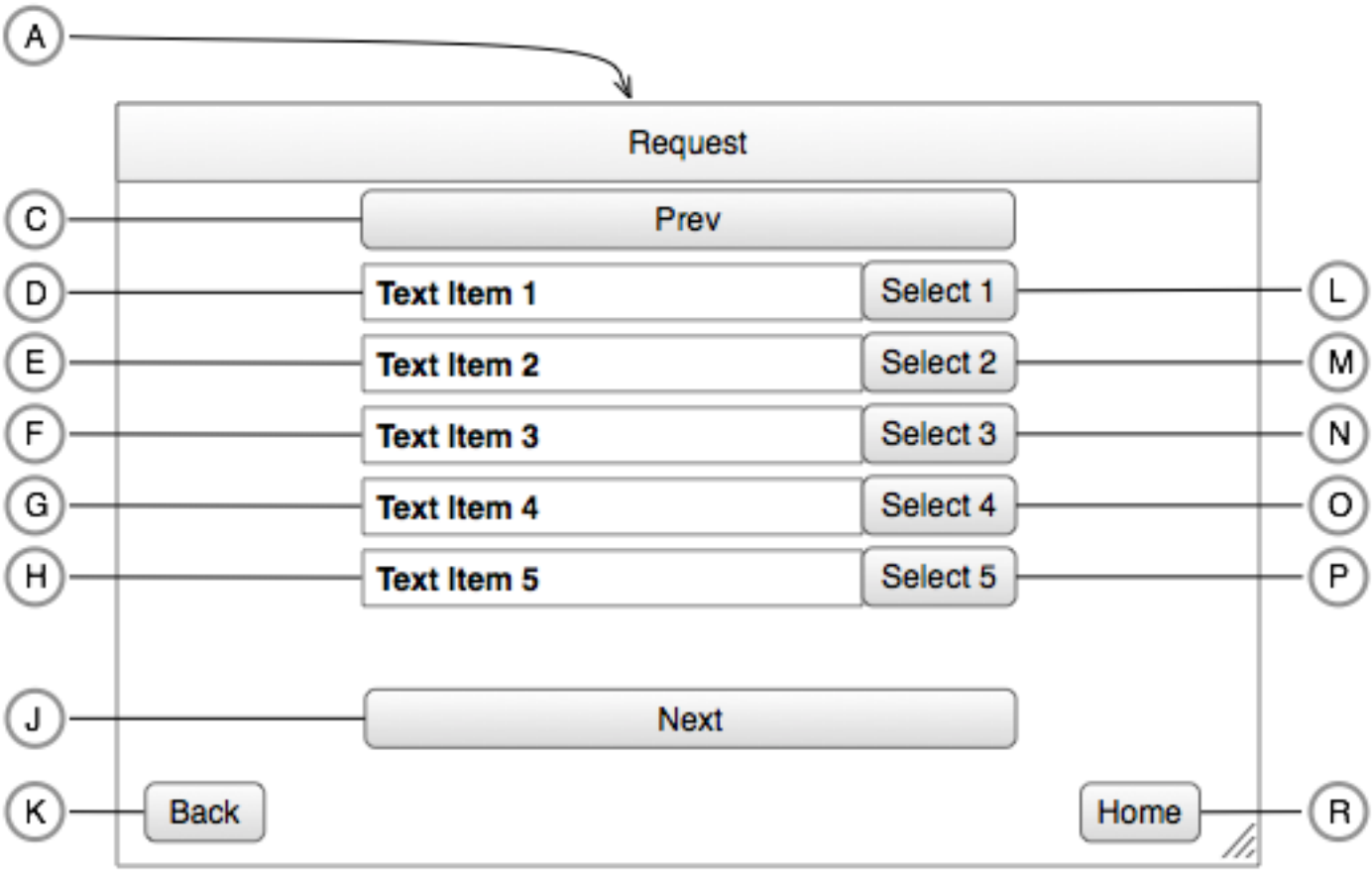


iPod Now Playing Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodNowPlayingScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].playlistName=”<value>”
E	N S[s].artistName=”<value>”
F	N S[s].albumName=”<value>”
G	N S[s].songName=”<value>”
H	N/A
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!KeyRelease Previous
M	EVENT C[c].Z[z]!KeyRelease Next
N	EVENT C[c].Z[z]!KeyRelease Plus
O	EVENT C[c].Z[z]!KeyRelease Minus
P	EVENT C[c].Z[z]!KeyRelease Stop
Q	EVENT C[c].Z[z]!KeyRelease Pause
R	N/A
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

iPod Request Screen

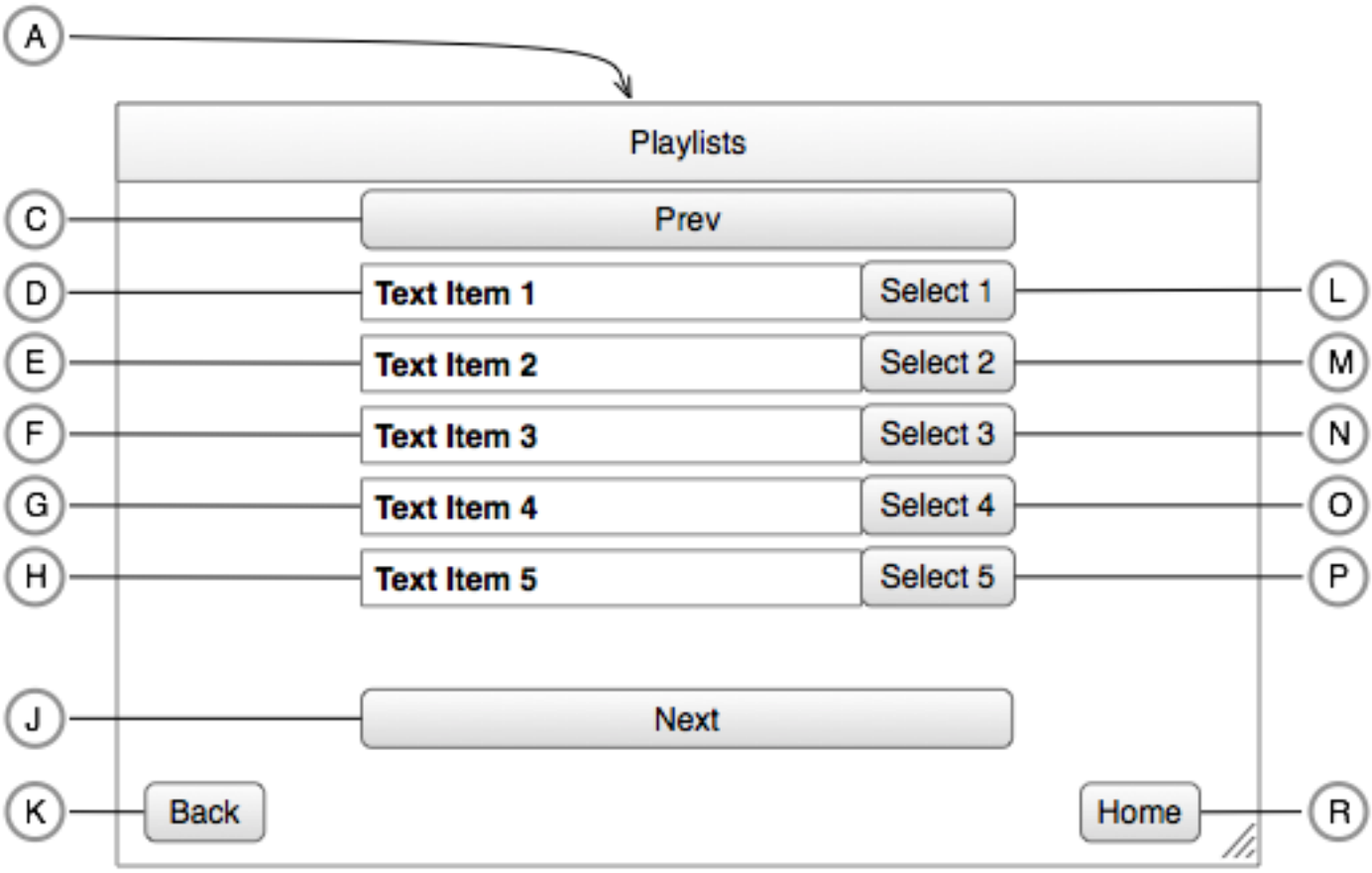


iPod Request Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodRequestScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

iPod Request by Playlist Screen

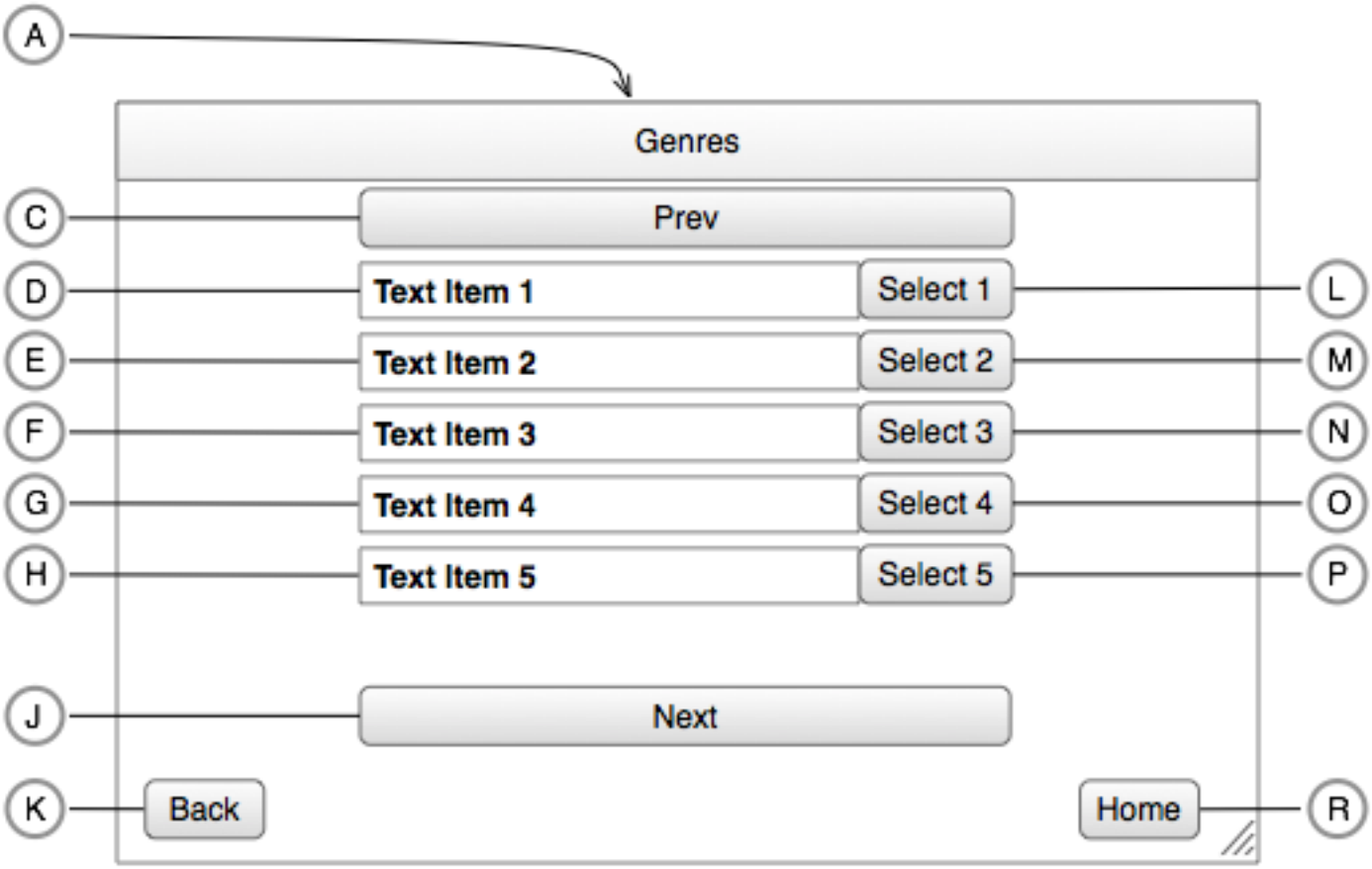


iPod Request by Playlist Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodPlaylistsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

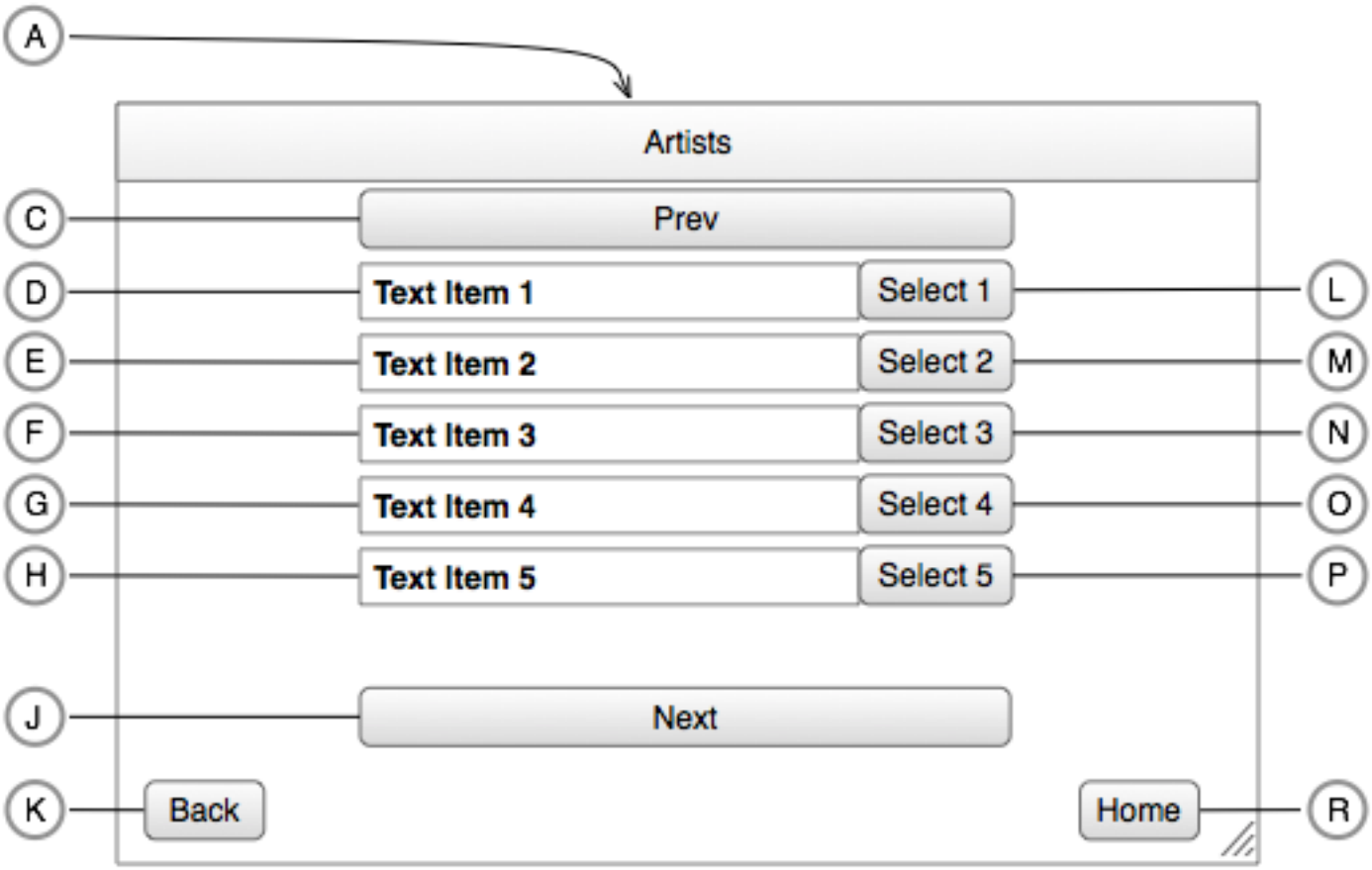
iPod Request by Genre



iPod Request by Genre Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodGenresScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A
V	N/A

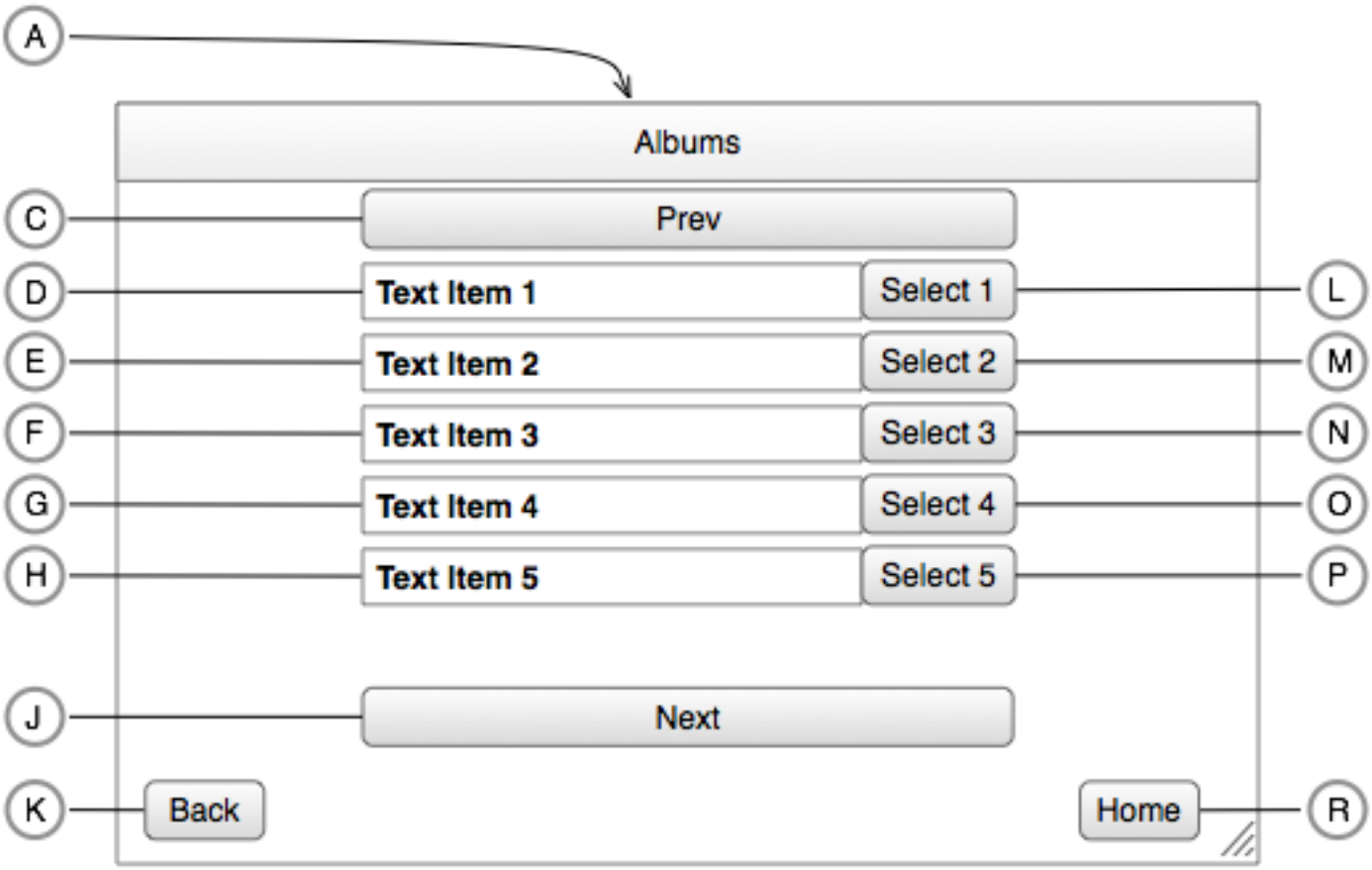
iPod Request by Artist Screen



iPod Request by Artist Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodArtistsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A
V	N/A

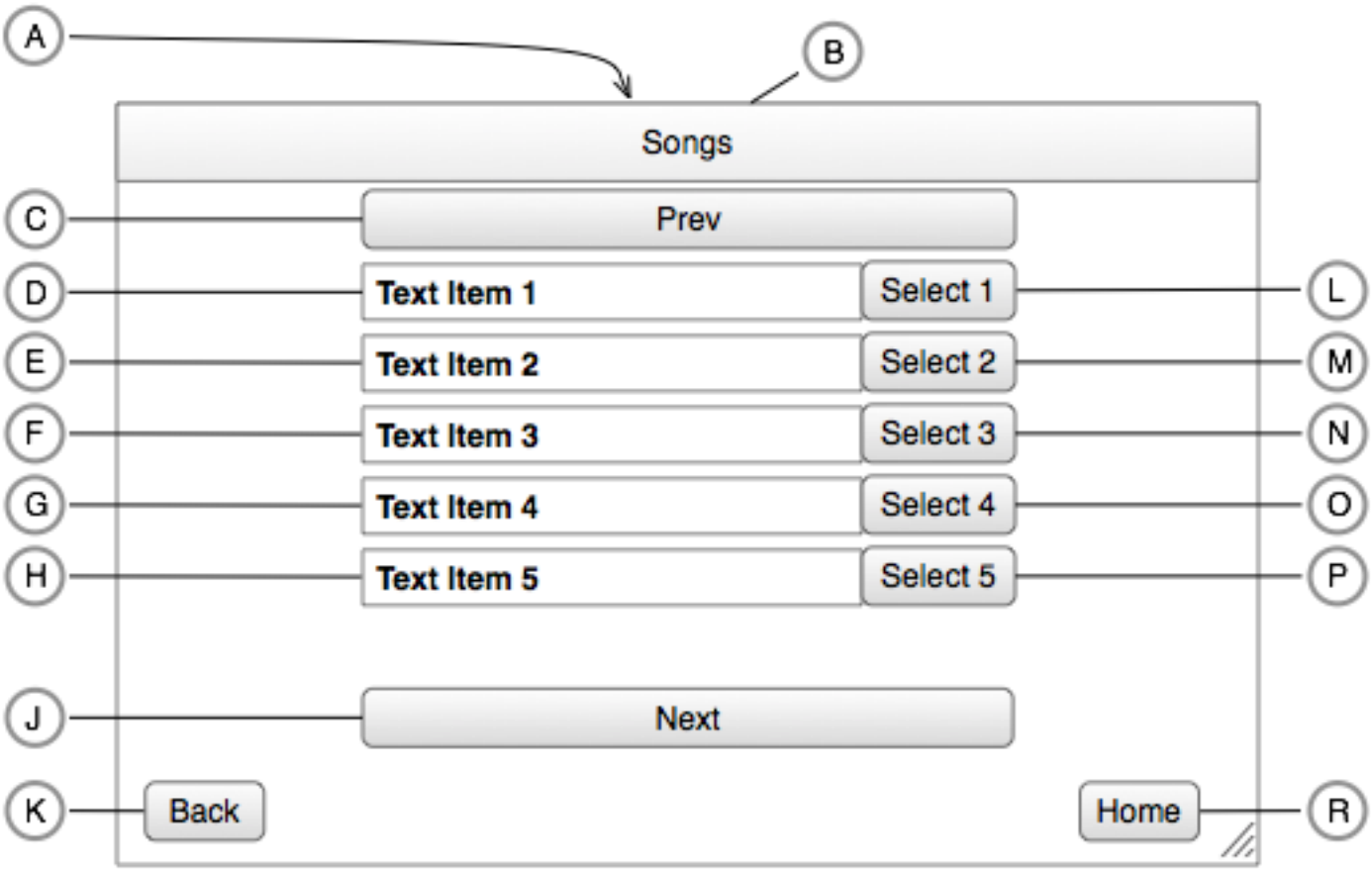
iPod Request by Album Screen



iPod Request by Album Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodAlbumsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A
V	N/A

iPod Request by Song Screen

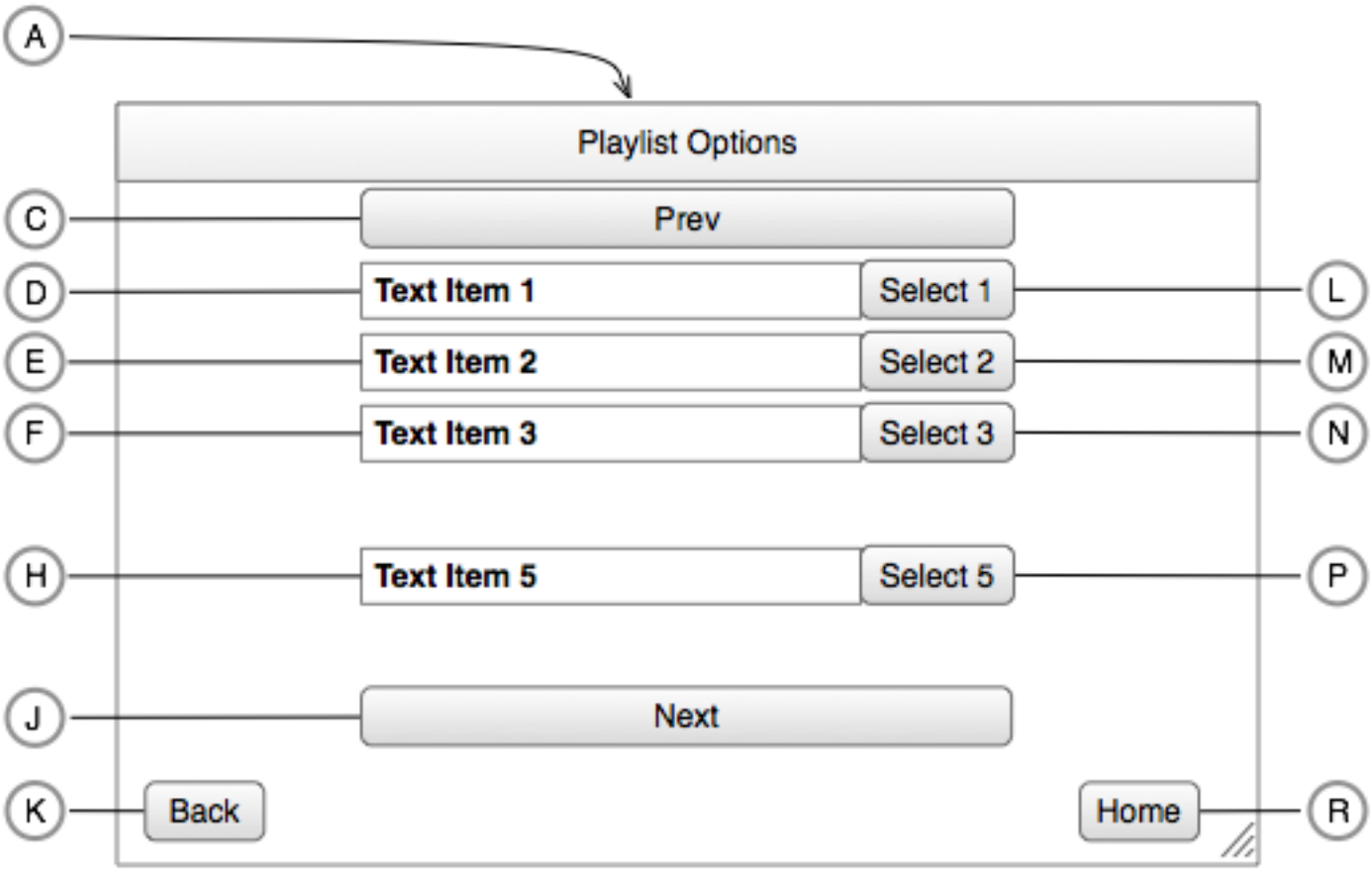


iPod Request by Song Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodSongsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

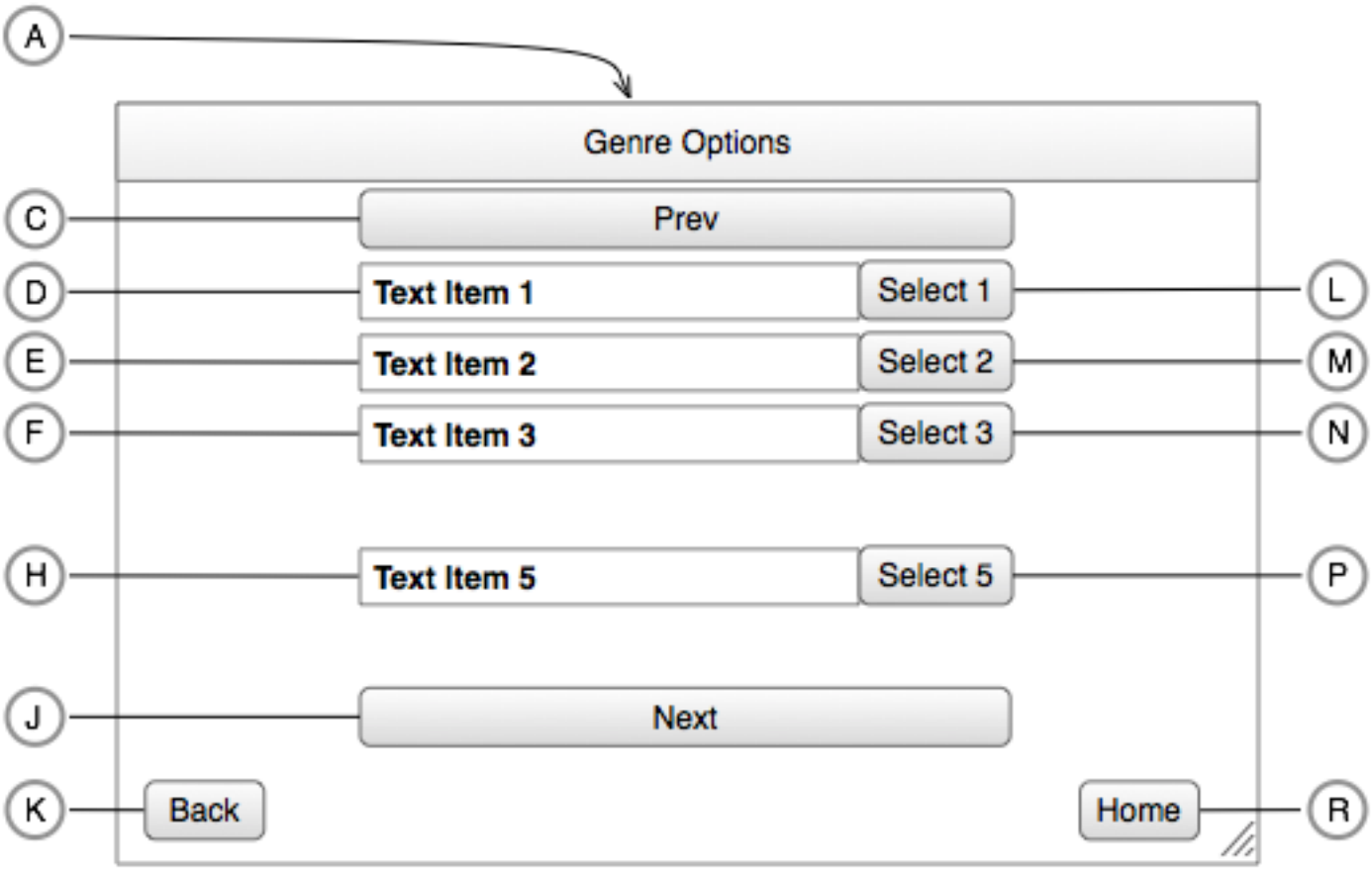
iPod Playlist Options Screen



iPod Playlist Options Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodPlaylistOptionsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N/A
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	N/A
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A
V	N/A

iPod Genre Options Screen

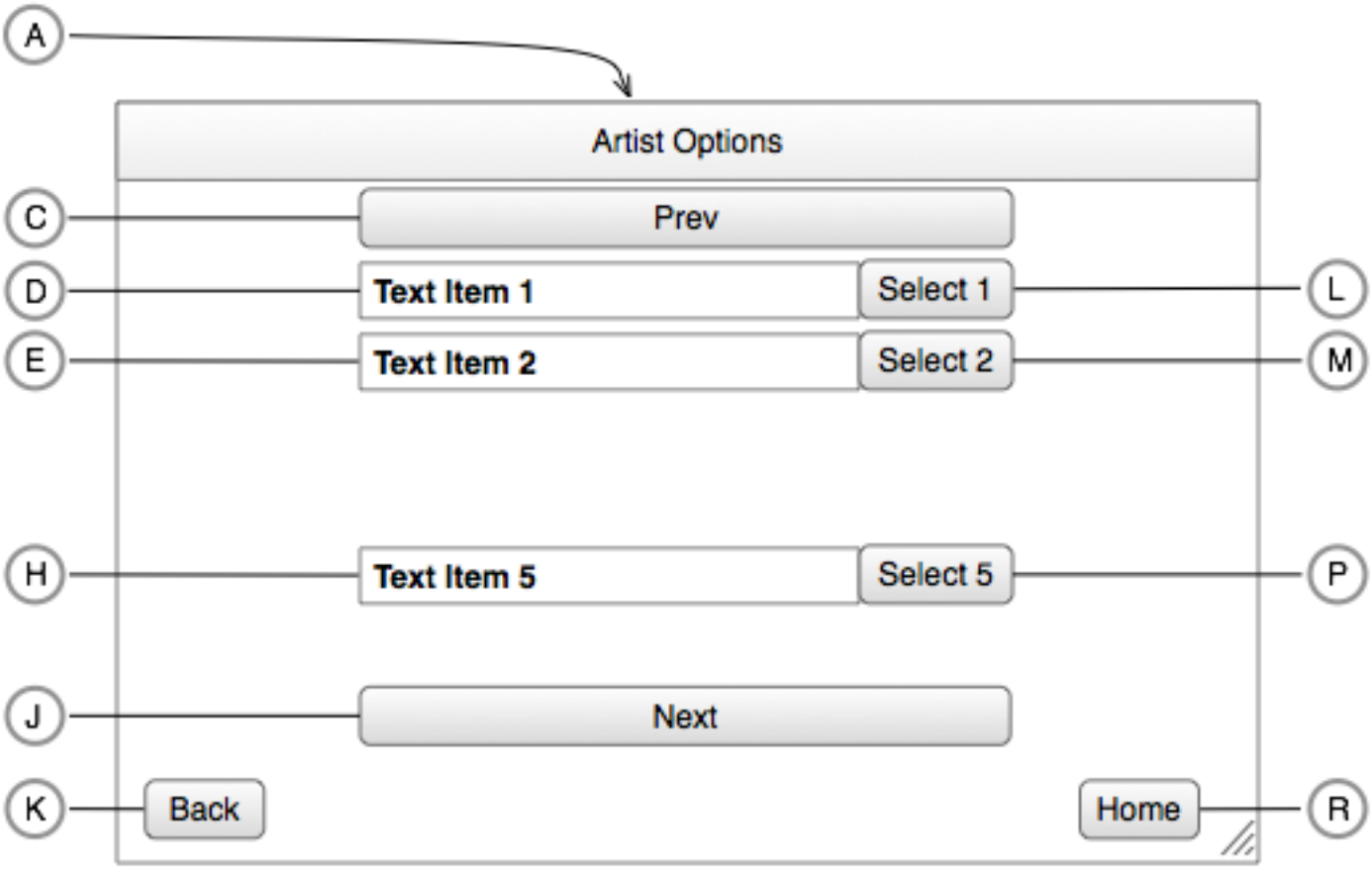


iPod Genre Options Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodGenreOptionsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N/A
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	N/A
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

iPod Artist Options Screen

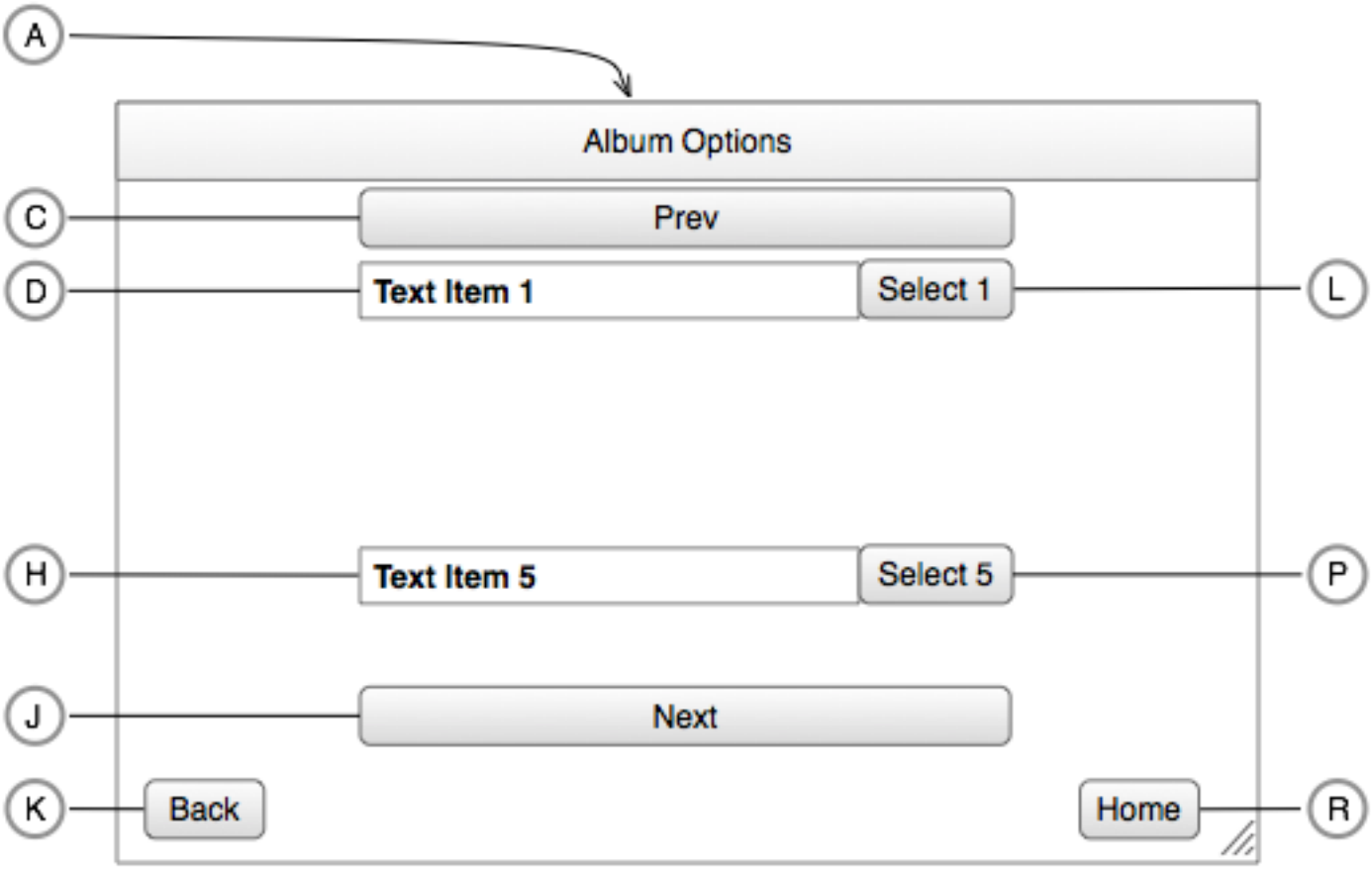


iPod Artist Options Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodArtistOptionsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N/A
G	N/A
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	N/A
O	N/A
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

iPod Album Options Screen

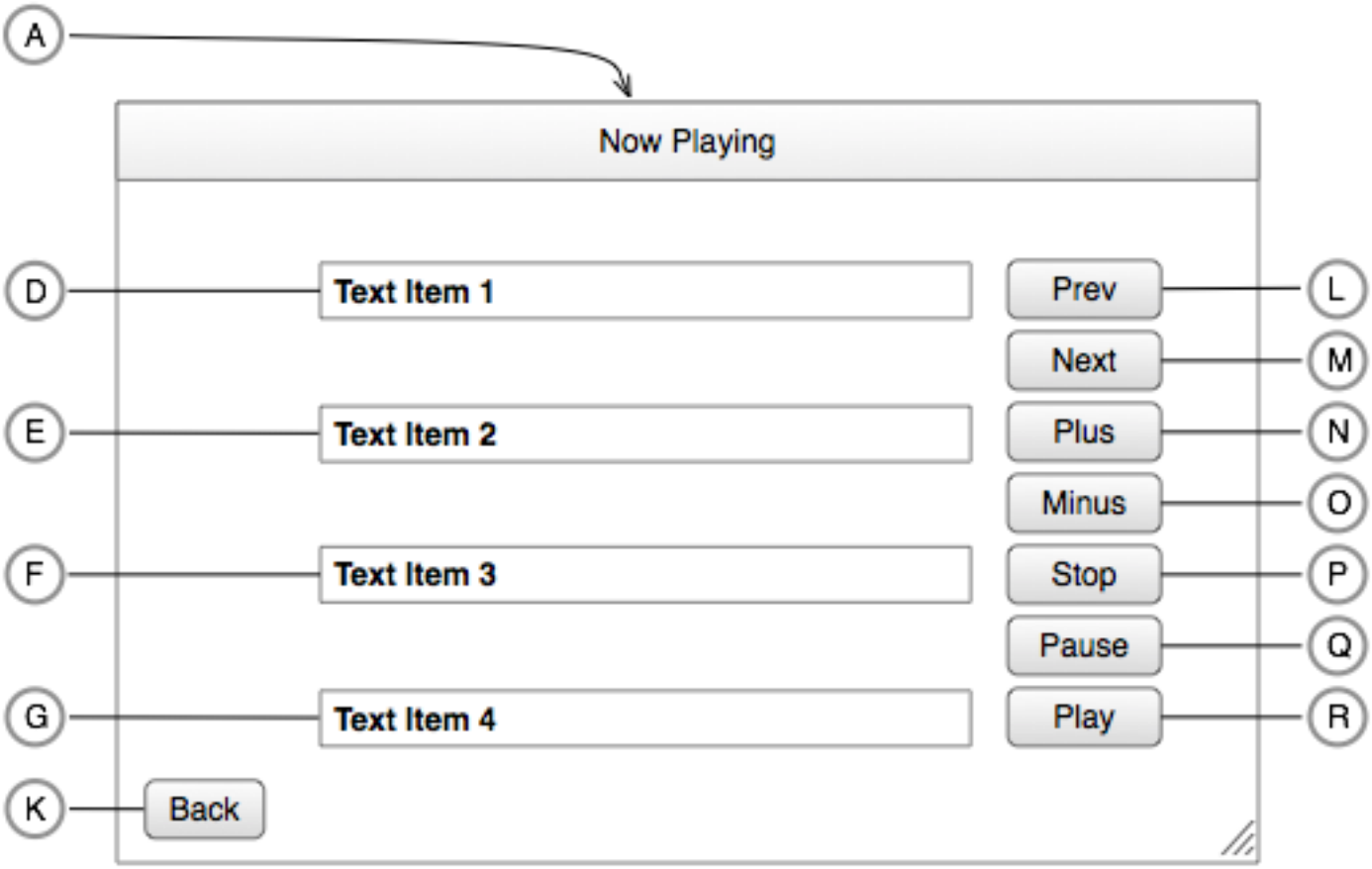


iPod Album Options Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”iPodAlbumOptionsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N/A
F	N/A
G	N/A
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	N/A
N	N/A
O	N/A
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

SMS3 Now Playing Screen

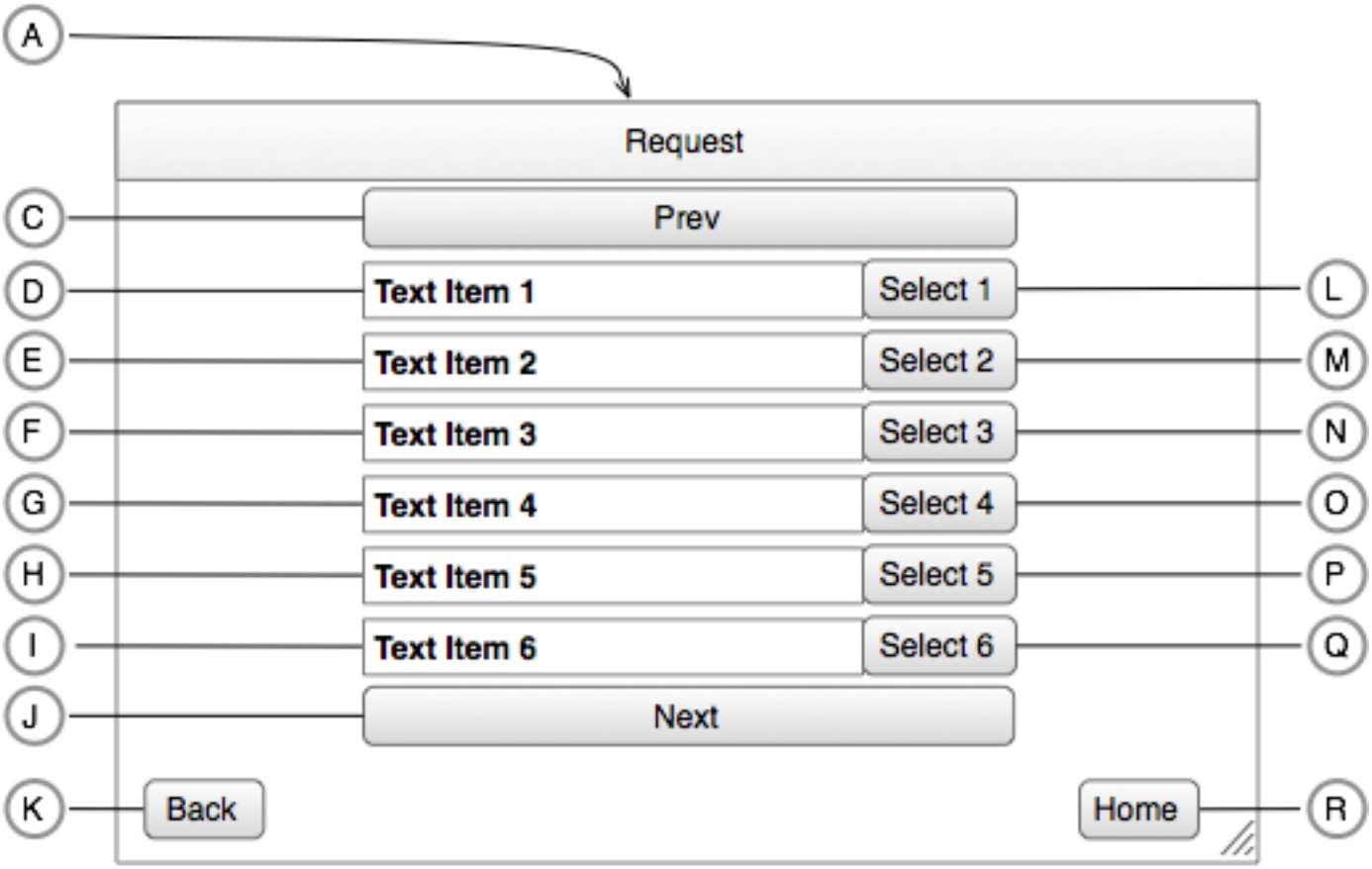


SMS3 Now Playing Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3NowPlayingScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].playlistName=”<value>”
E	N S[s].artistName=”<value>”
F	N S[s].albumName=”<value>”
G	N S[s].songName=”<value>”
H	N/A
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!KeyRelease Previous
M	EVENT C[c].Z[z]!KeyRelease Next
N	EVENT C[c].Z[z]!KeyRelease Plus
O	EVENT C[c].Z[z]!KeyRelease Minus
P	EVENT C[c].Z[z]!KeyRelease Stop
Q	EVENT C[c].Z[z]!KeyRelease Pause
R	N/A
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

SMS3 Request Screen

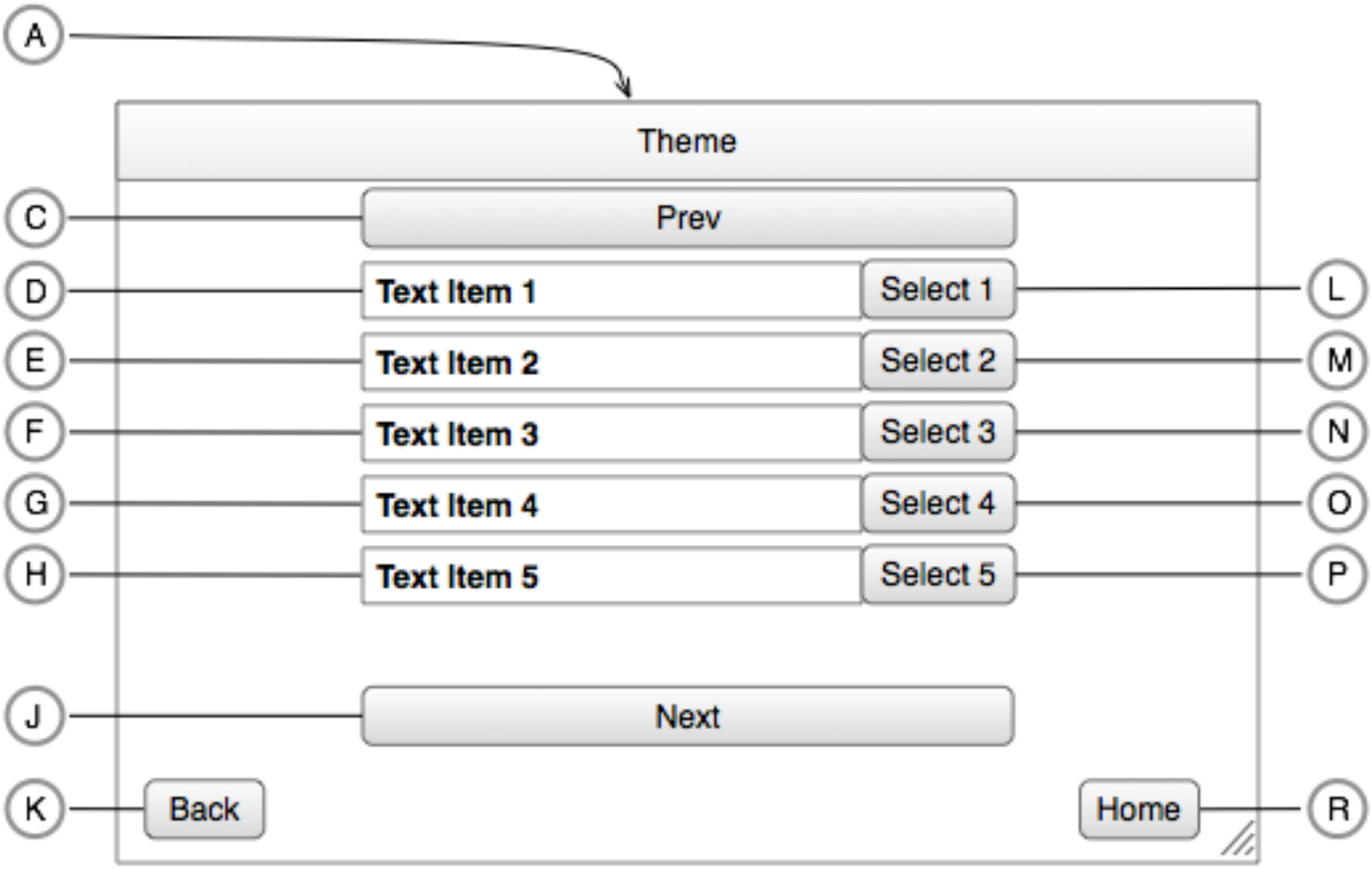


SMS3 Request Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3RequestScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N S[s].MMMenuItem[6].text=”<text string>”
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	EVENT C[c].Z[z]!MMSelectItem 6
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

SMS3 Request by Theme Screen

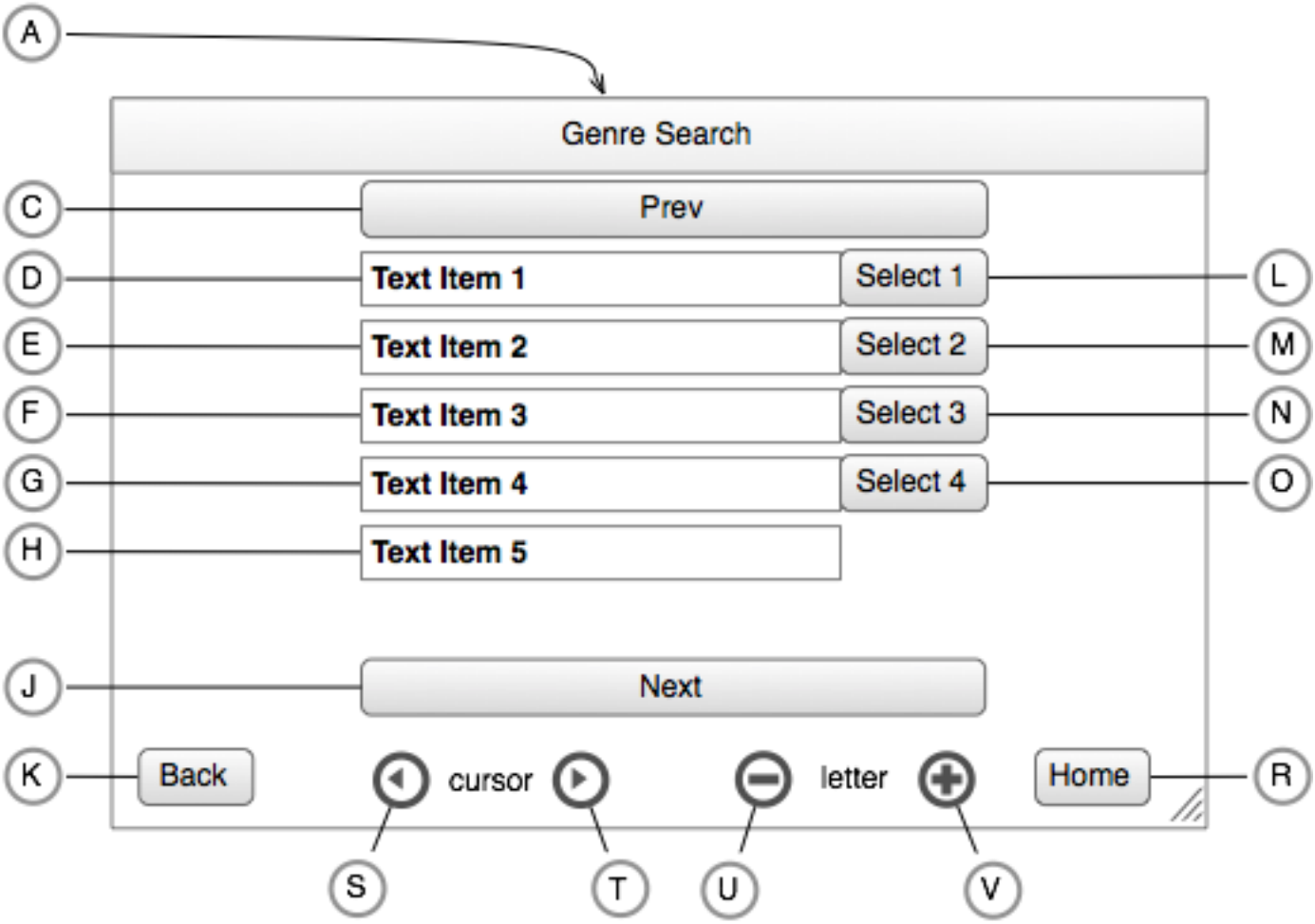


SMS3 Request by Theme Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3ThemesScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

SMS3 Request by Genre Screen

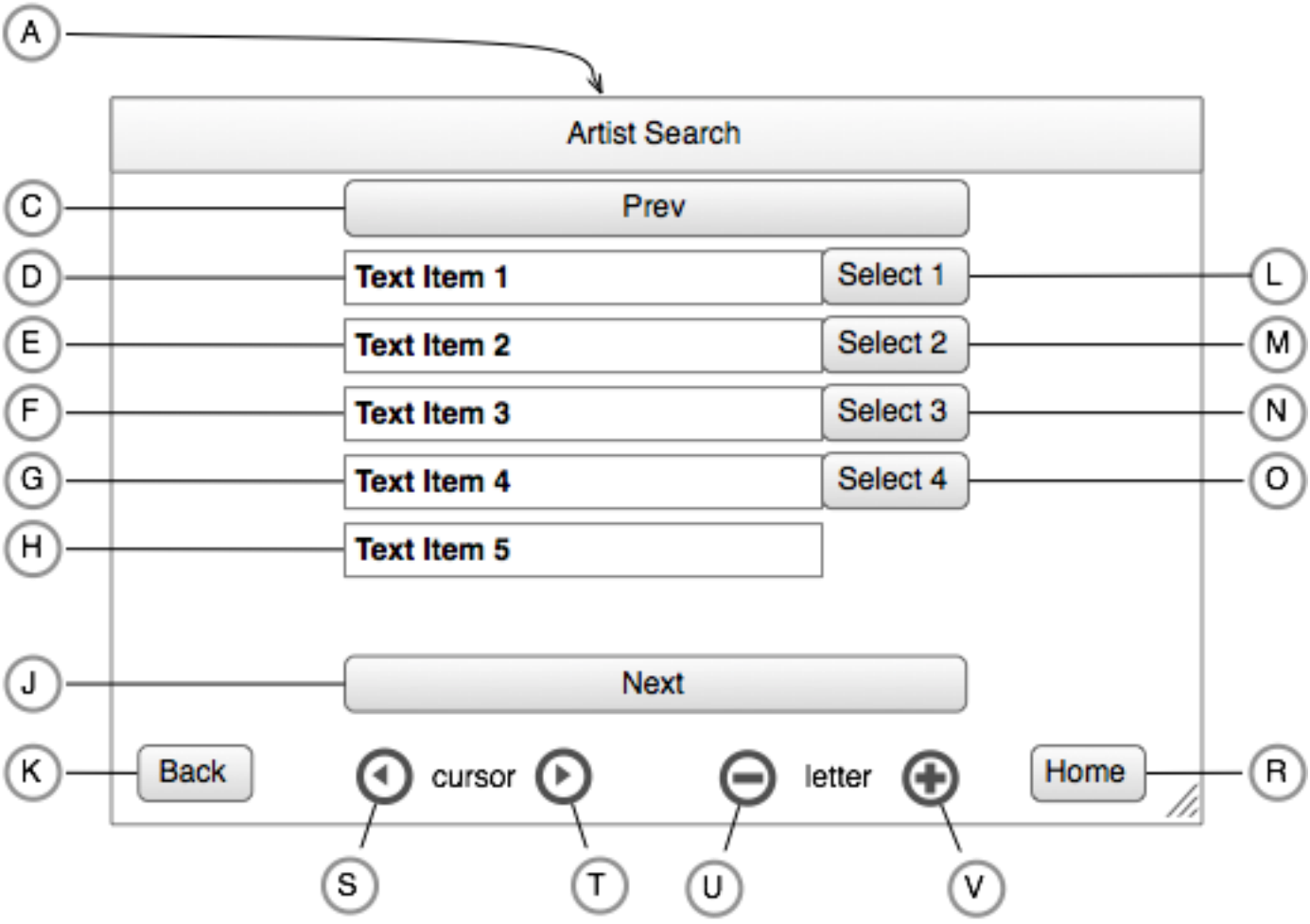


SMS3 Request by Genre Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3GenresScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	N/A
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	EVENT C[c].Z[z]!MMCursorPrev
T	EVENT C[c].Z[z]!MMCursorNext
U	EVENT C[c].Z[z]!MMLetterUp

Screen Designator	RIO Command/Response
V	EVENT C[c].Z[z]!MMLetterDown

SMS3 Request by Artist Screen

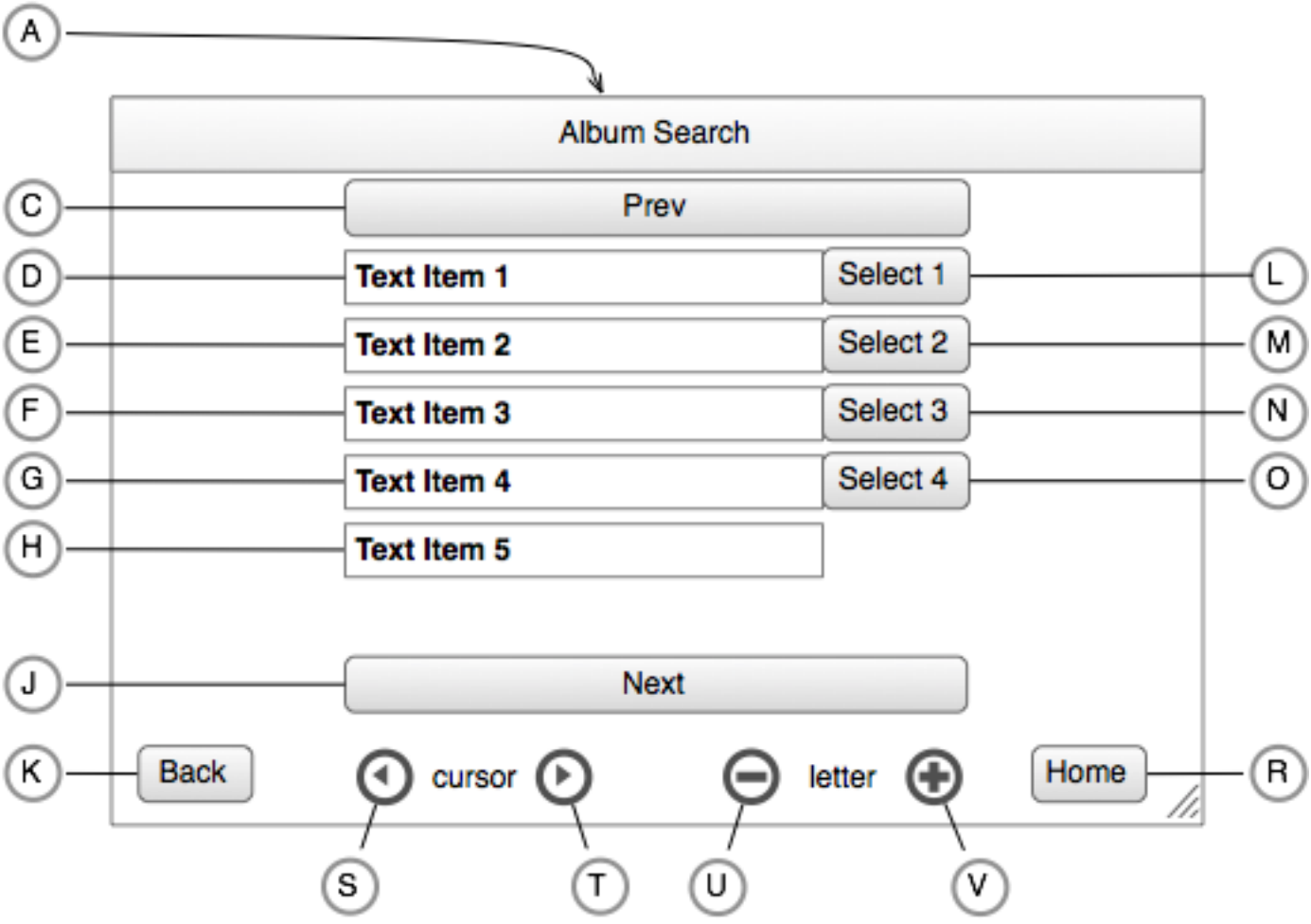


SMS3 Request by Artist Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3ArtistsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	N/A
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	EVENT C[c].Z[z]!MMCursorPrev
T	EVENT C[c].Z[z]!MMCursorNext
U	EVENT C[c].Z[z]!MMLetterUp

Screen Designator	RIO Command/Response
V	EVENT C[c].Z[z]!MMLetterDown

Request by Album Screen

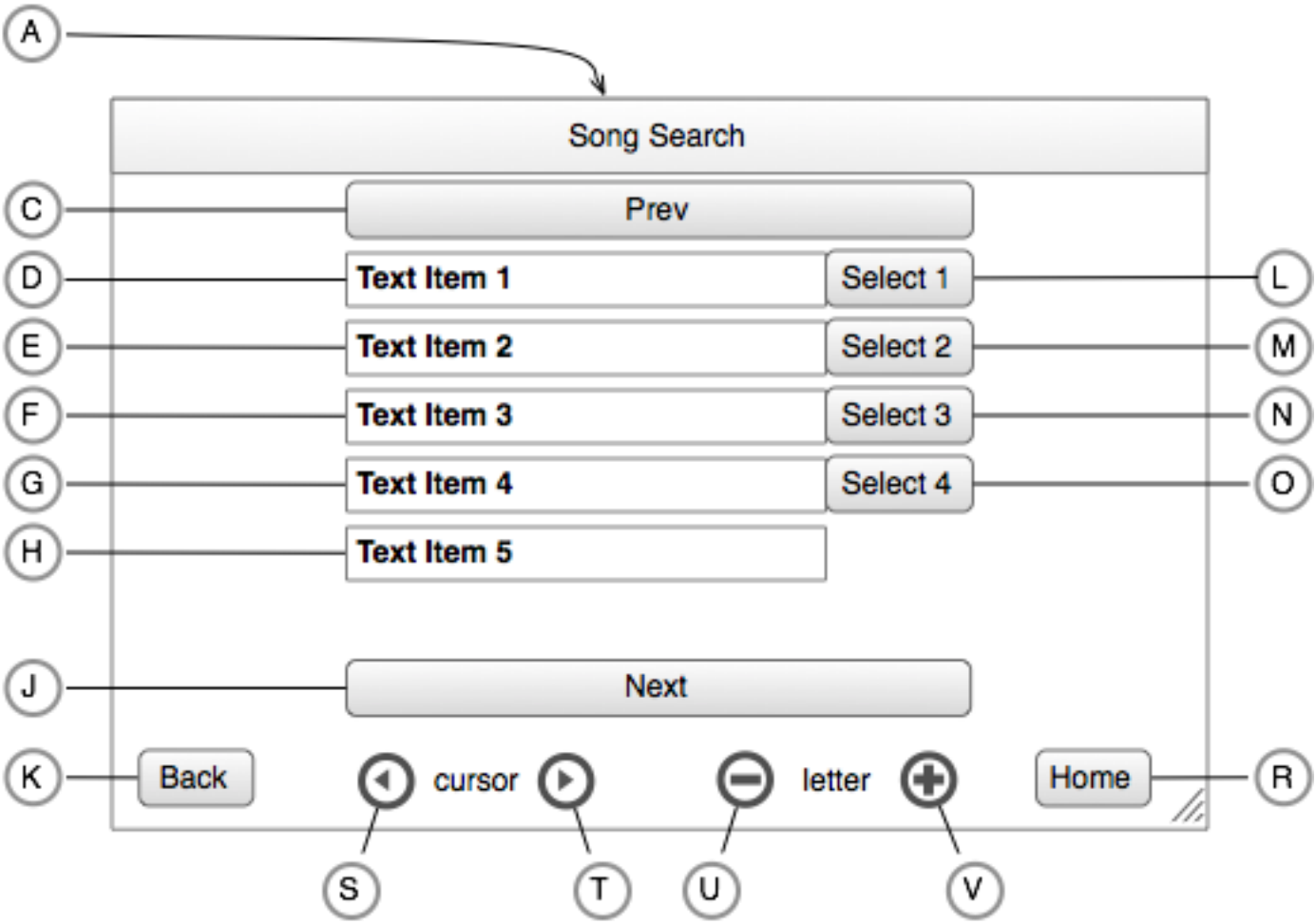


Request by Album Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3AlbumsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	N/A
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	EVENT C[c].Z[z]!MMCursorPrev
T	EVENT C[c].Z[z]!MMCursorNext
U	EVENT C[c].Z[z]!MMLetterUp

Screen Designator	RIO Command/Response
V	EVENT C[c].Z[z]!MMLetterDown

SMS3 Request by Song Title Screen

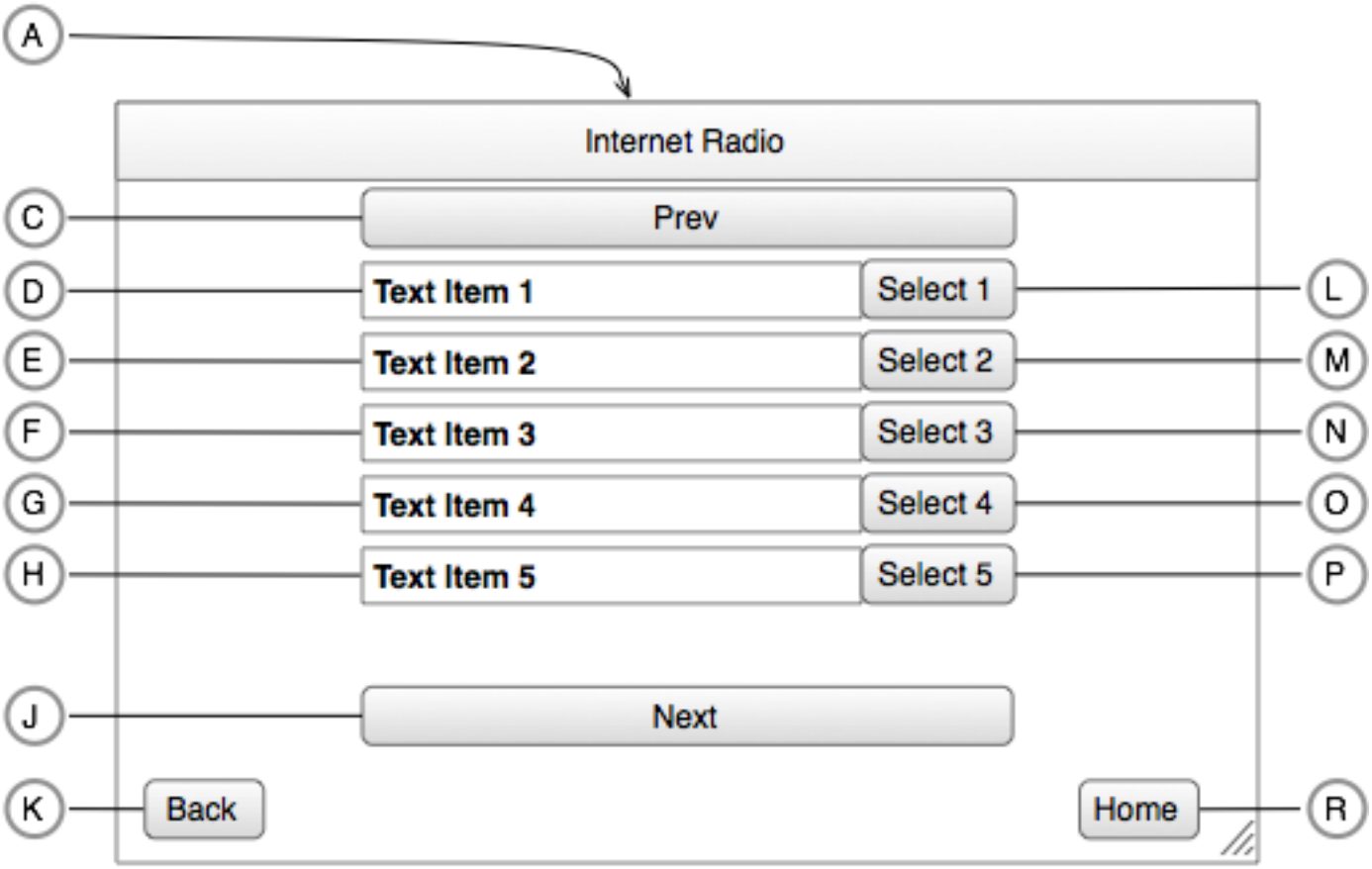


SMS3 Request by Song Title Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3SongsScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	N/A
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	EVENT C[c].Z[z]!MMCursorPrev
T	EVENT C[c].Z[z]!MMCursorNext
U	EVENT C[c].Z[z]!MMLetterUp

Screen Designator	RIO Command/Response
V	EVENT C[c].Z[z]!MMLetterDown

SMS3 Request by Internet Radio Screen

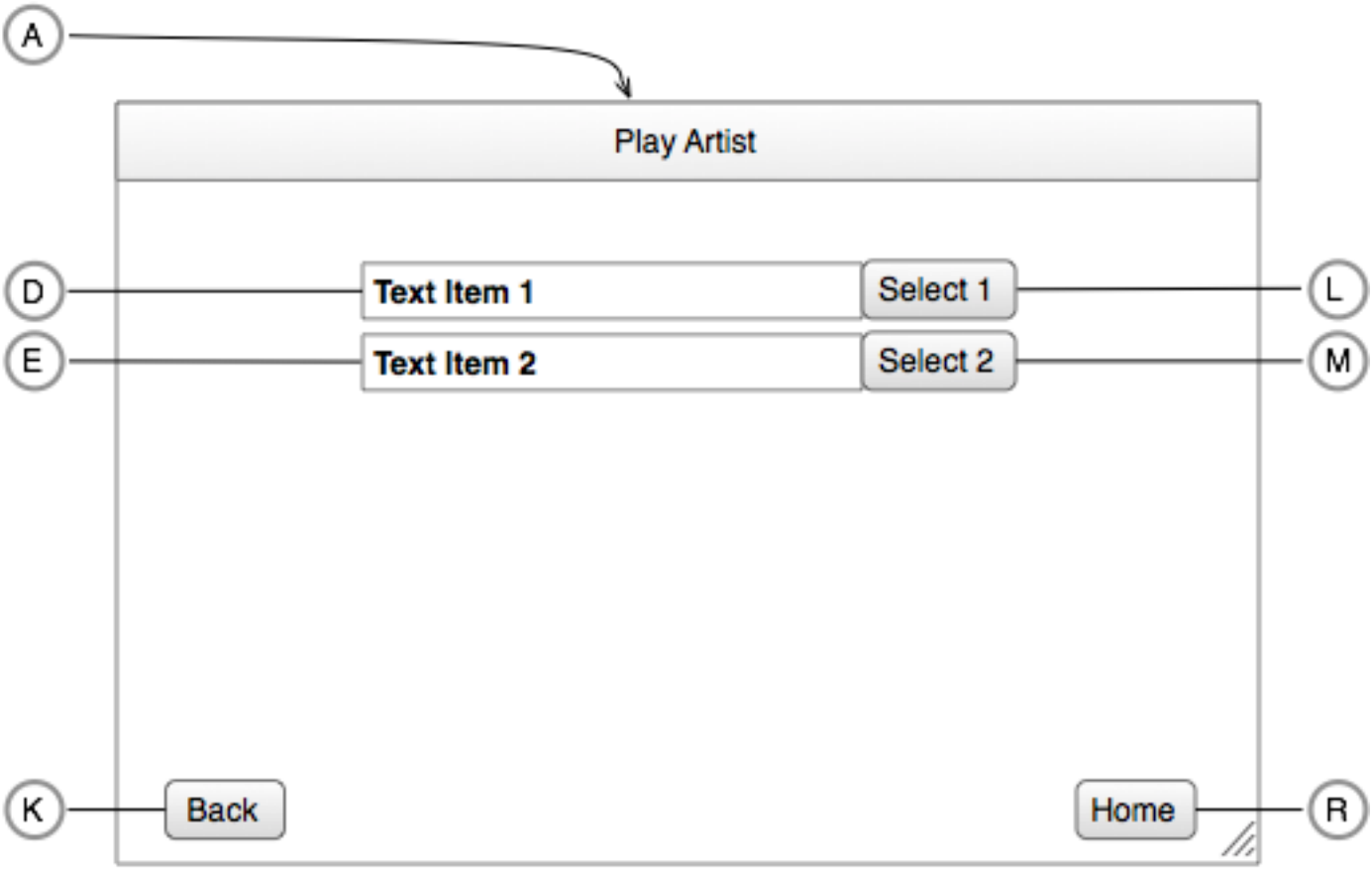


SMS3 Request by Internet Radio Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3InternetRadioScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

SMS3 Play Artist Screen

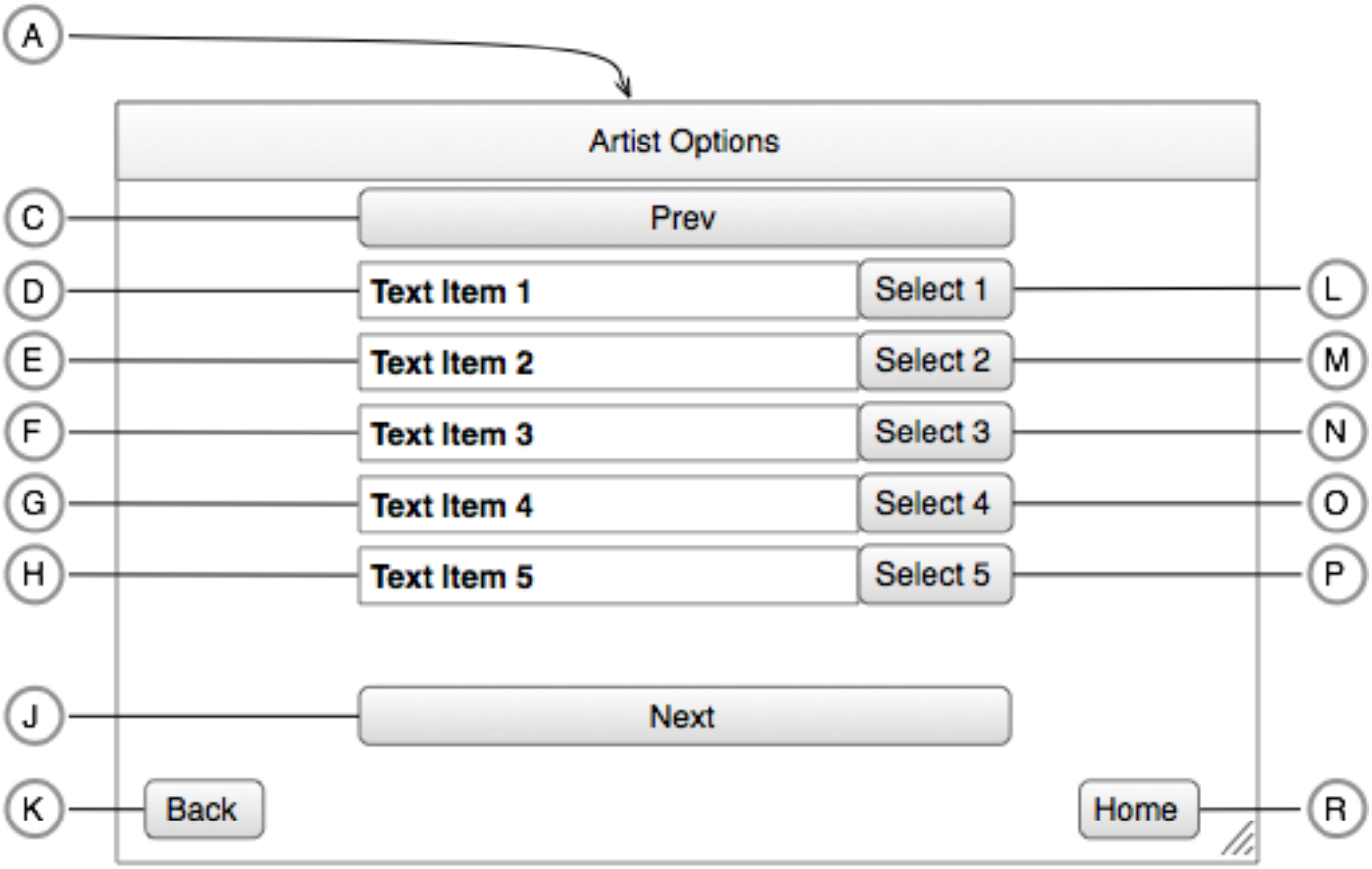


SMS3 Play Artist Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3PlayOptionsScreen”
B	N/A
C	N/A
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N/A
G	N/A
H	N/A
I	N/A
J	N/A
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	N/A
O	N/A
P	N/A
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

SMS3 Play Album by Artist Screen

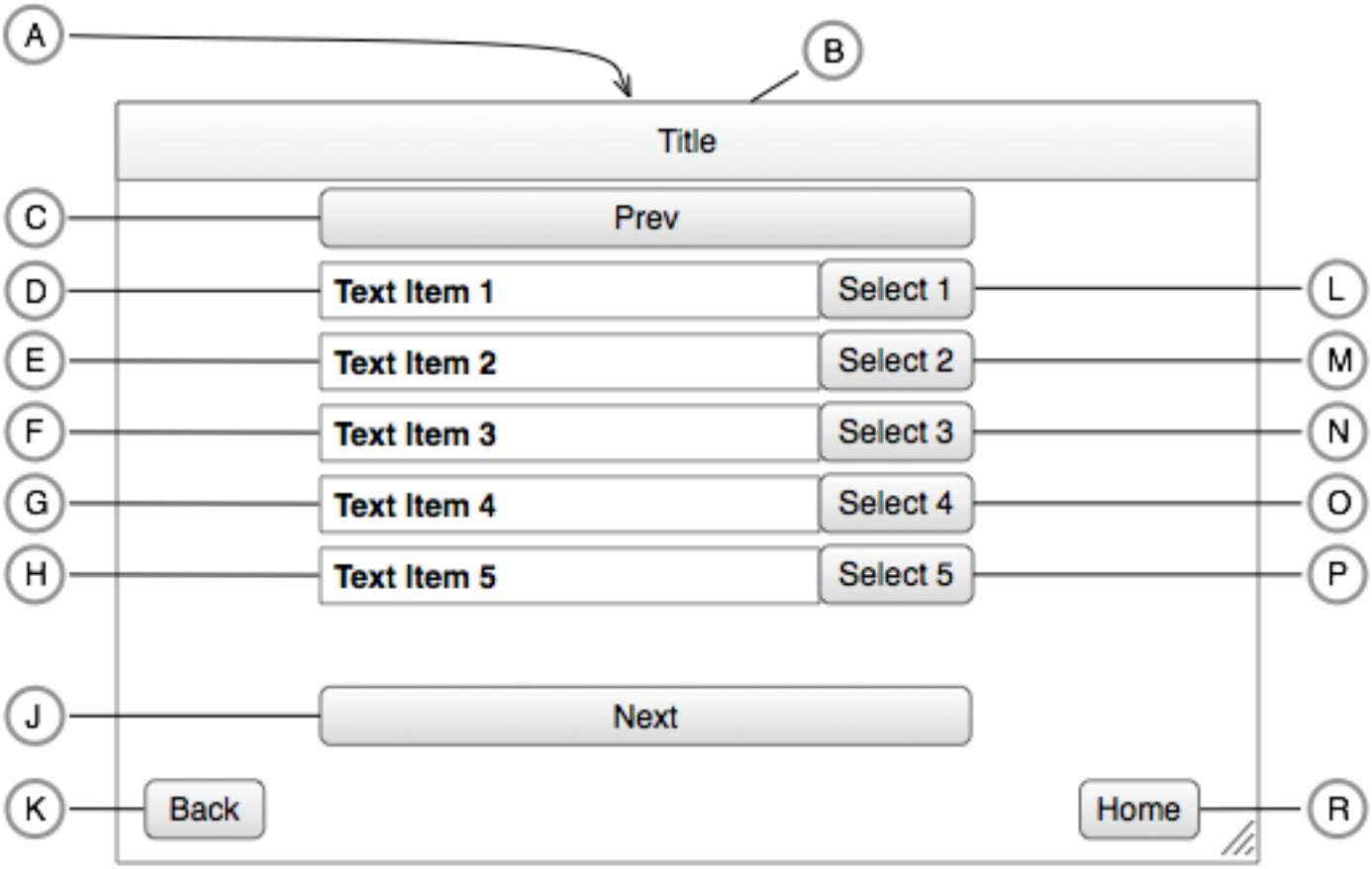


SMS3 Play Album by Artist Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SMS3AlbumsByArtistScreen”
B	N/A
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A

Screen Designator	RIO Command/Response
V	N/A

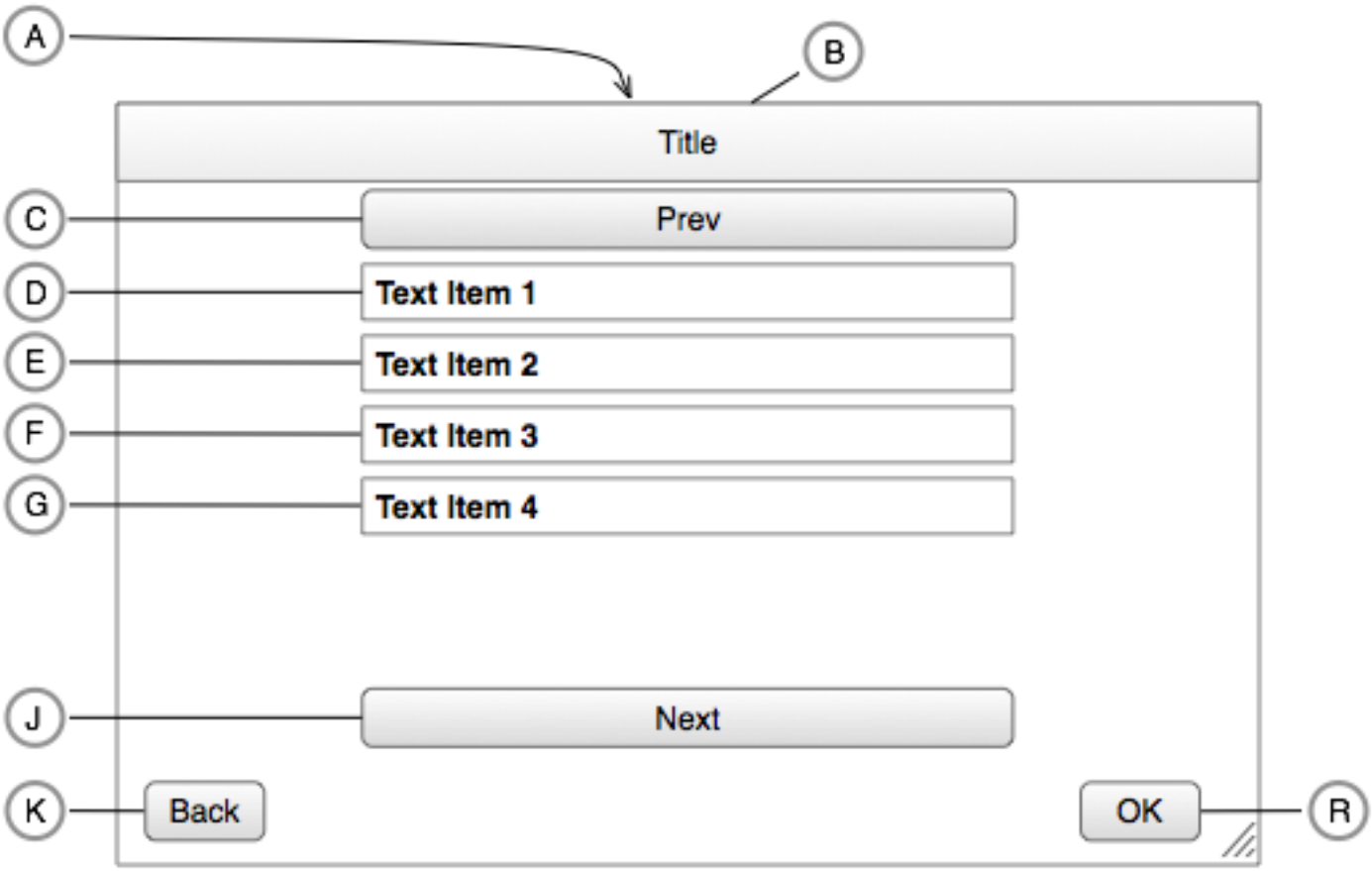
Menu Screen



Menu Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SourceMenuScreen”
B	N S[s].MMTitle.text=”<title string>”
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N S[s].MMMenuItem[5].text=”<text string>”
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMSelectItem 1
M	EVENT C[c].Z[z]!MMSelectItem 2
N	EVENT C[c].Z[z]!MMSelectItem 3
O	EVENT C[c].Z[z]!MMSelectItem 4
P	EVENT C[c].Z[z]!MMSelectItem 5
Q	N/A
R	EVENT C[c].Z[z]!MMInit
S	N/A
T	N/A
U	N/A
V	N/A

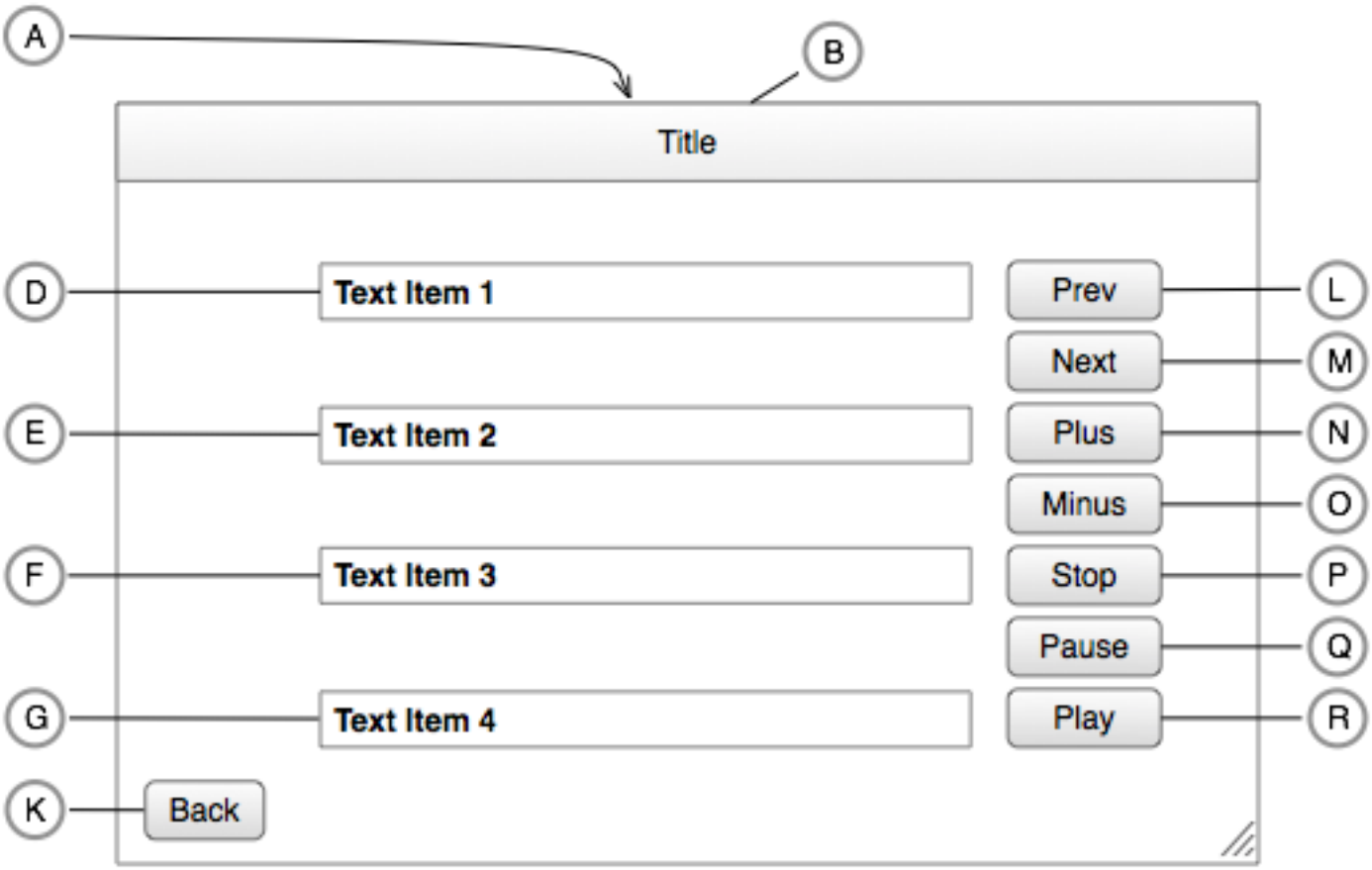
MM Info Screen



Info Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SourceInfoScreen”
B	N S[s].MMTitle.text=”<title string>”
C	EVENT C[c].Z[z]!MMPrevItems
D	N S[s].MMMenuItem[1].text=”<text string>”
E	N S[s].MMMenuItem[2].text=”<text string>”
F	N S[s].MMMenuItem[3].text=”<text string>”
G	N S[s].MMMenuItem[4].text=”<text string>”
H	N/A
I	N/A
J	EVENT C[c].Z[z]!MMNextItems
K	EVENT C[c].Z[z]!MMPrevScreen
L	N/A
M	N/A
N	N/A
O	N/A
P	N/A
Q	N/A
R	EVENT C[c].Z[z]!MMSelectOk
S	N/A
T	N/A
U	N/A
V	N/A

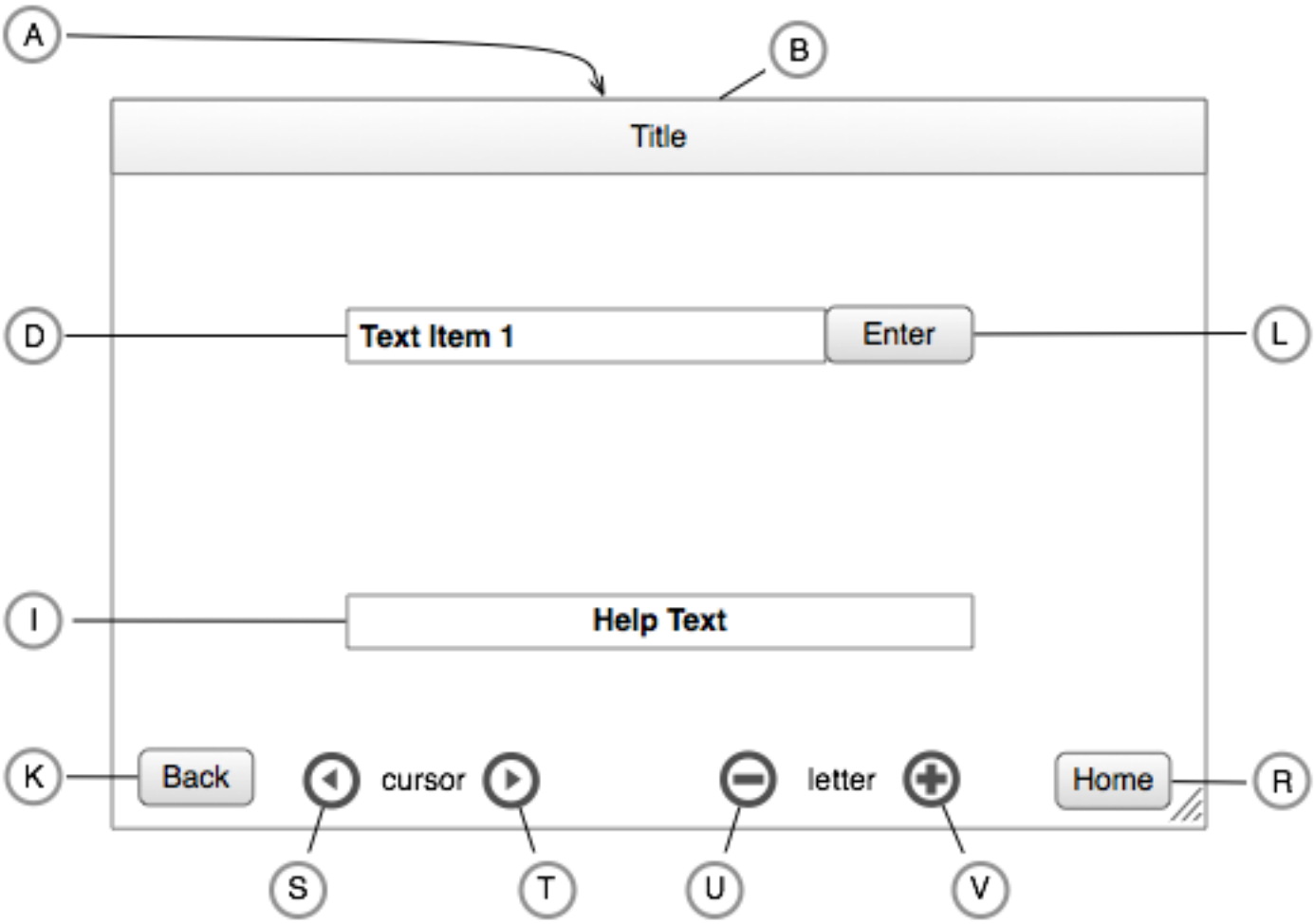
Now Playing Screen



Now Playing Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SourceNowPlayingScreen”
B	N S[s].MMTitle.text=”<title string>”
C	N/A
D	N S[s].playlistName=”<value>” / N S[s].playlistName=”<value>”
E	N S[s].artistName=”<value>”
F	N S[s].albumName=”<value>”
G	N S[s].songName=”<value>”
H	N/A
I	N/A
J	N/A
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!KeyRelease Previous
M	EVENT C[c].Z[z]!KeyRelease Next
N	EVENT C[c].Z[z]!KeyRelease Plus
O	EVENT C[c].Z[z]!KeyRelease Minus
P	EVENT C[c].Z[z]!KeyRelease Stop
Q	EVENT C[c].Z[z]!KeyRelease Pause
R	EVENT C[c].Z[z]!KeyRelease Play
S	N/A
T	N/A
U	N/A
V	N/A

Text Entry Screen



Text Entry Screen Elements

Screen Designator	RIO Command/Response
A	N S[s].MMScreen=”SourceTextEntryScreen”
B	N S[s].MMTitle.text=”<title string>”
C	N/A
D	N S[s].MMTextField.text=”<text string>”
E	N/A
F	N/A
G	N/A
H	N/A
I	N S[s].MMHelp.text=”<text string>”
J	N/A
K	EVENT C[c].Z[z]!MMPrevScreen
L	EVENT C[c].Z[z]!MMTextField "<text string>"
M	N/A
N	N/A
O	N/A
P	N/A
Q	N/A
R	N/A
S	EVENT C[c].Z[z]!MMCursorPrev
T	EVENT C[c].Z[z]!MMCursorNext
U	EVENT C[c].Z[z]!MMLetterUp
V	EVENT C[c].Z[z]!MMLetterDown

Media Management Long List support

Setting Menu Notification Details

Specify Menu Notification Format

EVENT C[c].Z[z]!MMFormat “[RIO | JSON]”

JSON Response Format

Field Descriptions

totalItems - the total number of items available in this menu

numItems - the number of items in this notification list

menuItems - an array of menu items containing these fields:

id - the instance of this menu item
text - the text to be displayed for this menu item

If the MMVerbosity is set to 2, these JSON element should be present:
• isFirst - if true, this is the first menu item in the menu
• isLast - if true, this is the last menu item in the menu
• isMenu - if true, this menu item is a menu selection

Example:


```
{
  "totalItems" : 100,
  "numItems" : 2,
  "menuItems" : [
    {
      "id" : 1,
      "text" : "Americana",
      "isFirst" : true,
      "isLast" : false,
      "isMenu" : true
    },
    {
      "id" : 2,
      "text" : "Angst",
      "isFirst" : false,
      "isLast" : false,
      "isMenu" : true
    }
  ]
}
```

Specify Menu Notification Verbosity

EVENT C[c].Z[z]!MMVerbosity [1 | 2]

Verbosity level 1 results in only the MMMenuItem.text notifications to be transmitted.

Verbosity level 2 results in all verbosity level 1 notifications as well as MMMenuItem.attr notifications.

Specify Menu Indexing Method

EVENT C[c].Z[z]!MMIndex “[RELATIVE | ABSOLUTE]”

Specify Menu Notification Max Items

This item is used to specify the maximum number of items that should be delivered in response to an MM Menu operation (MMInit, MMChooseItem, MMPrevScreen, MMPrevItems and MMNextItems). This setting is only relevant when MMIndex is set to ABSOLUTE. A setting of MMIndex as RELATIVE always results in 5 items returned.

EVENT C[c].Z[z]!MMMaxItems <1 to 256>

Navigate to a Menu Item by Index

This EVENT is used to navigate to a menu item that were received via notification. Subsequent MMPrevItems and MMNextItems will be made relative to the new menu position.

EVENT C[c].Z[z]!MMStartItem <1 to 2^32>

Navigate to a Menu Item by Character

This EVENT is used to navigate to a menu item that were received via notification by specifying the first character of the desired menu item. Subsequent MMPrevItems and MMNextItems will be made relative to the new menu position.

EVENT C[c].Z[z]!MMStartChar “<start character>”

Choosing a Menu Item

This EVENT is used to select from the menu items that were received via notification. Unlike the MMSelectItem EVENT, MMChooseItem provides ‘absolute’ access to all menu items in the current content.

EVENT C[c].Z[z]!MMChooseItem <1 to 2^32>

Menu Item Notifications

Text

N S[s].MMMenuItem[1 to 2^32].text=”<text string>”

Attributes

N S[s].MMMenuItem[1 to 2^32].attr=”<text string>”

Attribute types:
M - This menu item is a menu selection
E - This is the last menu item in this transaction
L - This is the last menu item in this menu context

N S[s].MMMenu.totalItems=””
N S[s].MMMenu.numItems=””

More on the ‘key’ string...

The key string is formatted as a dot-separated (‘.’) series of strings that refer to a hierarchical set of branches, tables and leafs.

Key strings are case insensitive.

Branches, represented as a capitalized string, serve to organize the system parameters by category. A key string can contain multiple branches.

For example, in the WATCH SYSTEM command response,

N System.status=”<value>”

‘System’ is a branch string.

Tables, represented by a capitalized string followed by a bracketed 1-based number, allow for instances of items such as controllers, zones, source, etc to be referenced by index. A key string can contain multiple tables.

For example, in the WATCH Source command response,

N S[1].type=”<value>”

‘S[s]’ is a table.

Leafs, represented by a lowercase ‘camel’ string and always the last string in the key string, refer to a specific system parameter. A key string has only one leaf string.

For example, in the WATCH Zone command response,

N C[1].Z[1].bass=”10”

‘bass’ is a leaf string.

Using PuTTY as a RIO Client

PuTTY is a popular free implementation of Telnet and SSH for Win32 and Unix platforms. It is possible to use PuTTY as a RIO client for testing purposes, provided it is configured correctly.

PuTTY is available here:

http://www.chiark.greenend.org.uk/~sgtatham/putty/

When connected via IP, you may use PuTTY in ‘Raw’ mode or ‘Telnet’ mode, depending on your testing needs.

With either mode, specify ‘Port’ as ‘9621’ in order for PuTTY to operate as a RIO client.

If you select ‘Raw’ as your ‘Connection type’, PuTTY will interoperate with a RIO compliant Russound device without any further configuration.

If you wish to use ‘Telnet’ mode, you must set ‘Category/Connection/Telnet/Telnet negotiation mode’ to ‘Passive’.

Low Power Considerations

RIO devices designed with low-power standby capability will enter a standby mode after approximately 5 to 7 minutes of inactivity (all zones and Home Theater Trigger are off), depending on the device. Third Party RIO clients need to make special considerations to assure uninterrupted operation.

RIO Keepalive

A RIO device with low-power standby capability will not enter standby if it receives a RIO command before the 5 to 7 minute standby time expires. Receiving a RIO command makes the RIO device aware that a RIO client device is present and in use.

The simplest way to keep a RIO device from entering a standby mode is to send an 'empty' RIO command to the RIO device at a time interval that is shorter than the 5 to 7 minute timeout period. An empty RIO command is merely a <CR> (0x0D hex) with no other characters in the message. The RIO device discards the command (no response is sent to the RIO client), but it is sufficient for keeping the RIO device from entering standby mode.

Waking a RIO device using Wake On LAN

RIO devices with low-power standby and IP capabilities are able to exit standby mode upon receiving a 'Wake-On-LAN' packet, known as a 'magic packet', over it's Ethernet port. For details on Wake-On-LAN, and formatting/generating a magic packet, go to http://en.wikipedia.org/wiki/Wake-on-LAN.

Constructing the correct magic packet requires knowledge of the IP and MAC Address of the RIO device under control. This information is available via the RIO 'GET' command. See the section ***RIO Protocol Syntax - The 'GET' Command*** for more details.