Language Reference Guide

Standard NetLinx API (SNAPI)
v1.14 - Devices

last revised: 1-11-2011

Software License and Warranty Agreement

- LICENSE GRANT. AMX grants to Licensee the non-exclusive right to use the AMX Software in the manner described in this License. The AMX Software is licensed, not sold. This license does not grant Licensee the right to create derivative works of the AMX Software. The AMX Software consists of generally available programming and development software, product documentation, sample applications, tools and utilities, and miscellaneous technical information. Please refer to the README.TXT file on the compact disc or download for further information regarding the components of the AMX Software. The AMX Software is subject to restrictions on distribution described in this License Agreement. AMX Dealer, Distributor, VIP or other AMX authorized entity shall not, and shall not permit any other person to, disclose, display, loan, publish, transfer (whether by sale, assignment, exchange, gift, operation of law or otherwise), license, sublicense, copy, or otherwise disseminate the AMX Software. Licensee may not reverse engineer, decompile, or disassemble the AMX Software.
- ACKNOWLEDGEMENT. You hereby acknowledge that you are an authorized AMX dealer, distributor, VIP or other AMX authorized entity in good standing and have the right to enter into and be bound by the terms of this Agreement.
- INTELLECTUAL PROPERTY. The AMX Software is owned by AMX and is protected by United States copyright laws, patent laws, international treaty provisions, and/or state of Texas trade secret laws. Licensee may make copies of the AMX Software solely for backup or archival purposes. Licensee may not copy the written materials accompanying the AMX Software.
- TERMINATION. AMX RESERVES THE RIGHT, IN ITS SOLE DISCRETION, TO TERMINATE THIS LICENSE FOR ANY REASON UPON WRITTEN NOTICE TO LICENSEE. In the event that AMX terminates this License, the Licensee shall return or destroy all originals and copies of the AMX Software to AMX and certify in writing that all originals and copies have been returned or destroyed.
- PRE-RELEASE CODE. Portions of the AMX Software may, from time to time, as identified in the AMX Software, include PRE-RELEASE CODE and such code may not be at the level of performance, compatibility and functionality of the GA code. The PRE-RELEASE CODE may not operate correctly and may be substantially modified prior to final release or certain features may not be generally released. AMX is not obligated to make or support any PRE-RELEASE CODE. ALL PRE-RELEASE CODE IS PROVIDED "AS IS" WITH NO WARRANTIES.
- LIMITED WARRANTY. AMX warrants that the AMX Software (other than pre-release code) will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt. AMX DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH REGARD TO THE AMX SOFTWARE. THIS LIMITED WARRANTY GIVES LICENSEE SPECIFIC LEGAL RIGHTS. Any supplements or updates to the AMX SOFTWARE, including without limitation, any (if any) service packs or hot fixes provided to Licensee after the expiration of the ninety (90) day Limited Warranty period are not covered by any warranty or condition, express, implied or statutory.
- LICENSEE REMEDIES. AMX's entire liability and Licensee's exclusive remedy shall be repair or replacement of the AMX Software that does not meet AMX's Limited Warranty and which is returned to AMX in accordance with AMX's current return policy. This Limited Warranty is void if failure of the AMX Software has resulted from accident, abuse, or misapplication. Any replacement AMX Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. Outside the United States, these remedies may not available. NO LIABILITY FOR CONSEQUENTIAL DAMAGES. IN NO EVENT SHALL AMX BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS AMX SOFTWARE, EVEN IF AMX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES/COUNTRIES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO LICENSEE.
- U.S. GOVERNMENT RESTRICTED RIGHTS. The AMX Software is provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph ©(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs ©(1) and (2) of the Commercial Computer Software Restricted Rights at 48 CFR 52.227-19, as applicable.
- SOFTWARE AND OTHER MATERIALS FROM AMX.COM MAY BE SUBJECT TO EXPORT CONTROL. The United States Export Control laws prohibit the export of certain technical data and software to certain territories. No software from this Site may be downloaded or exported (i) into (or to a national or resident of) Cuba, Iraq, Libya, North Korea, Iran, Syria, or any other country to which the United States has embargoed goods; or (ii) anyone on the United States Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Deny Orders. AMX does not authorize the downloading or exporting of any software or technical data from this site to any jurisdiction prohibited by the United States Export Laws.

This Agreement replaces and supersedes all previous AMX Software License Agreements and is governed by the laws of the State of Texas, and all disputes will be resolved in the courts in Collin County, Texas, USA. For any questions concerning this Agreement, or to contact AMX for any reason, please write: AMX License and Warranty Department, 3000 Research Drive, Richardson, TX 75082.

Table of Contents

Overview 1
- Devices 1
- Ports 1
- Channels 1
- Levels 2
- Commands 2
- SNAPI.axi 2

Programming 3
- Channels 3
- Ramping Channels 4
- Levels 4
- Commands 5
- General 5

Channel and Level Ranges 5

Commands and Escape Characters 6

Amplifier 9
- Amplifier State Charts 11

Audio Conferencer 13
- Audio Conferencer State Charts 24

Audio Mixer 25

Audio Processor 31
- Audio Processor State Charts 35

Audio Tape 37
- Audio Tape State Charts 40

Audio Tuner Device 43
- Audio Tuner Device State Charts 48

Camera 49
Camera State Charts 53
Digital Media Decoder 55
Digital Media Decoder State Charts 66
Digital Media Encoder 67
Digital Media Encoder State Charts 72
Digital Media Server 73
Digital Media Server State Charts 88
Digital Satellite System 91
Digital Satellite System State Charts 101
Digital Video Recorder 103
Digital Video Recorder State Charts 117
Disc Device 119
Disc Device State Charts 129
Document Camera 131
Document Camera State Charts 135
HVAC 137
HVAC Listener State Charts 141
IO Device 143
Keypad 145
Keypad Listener State Charts 146
Light 149
Light State Charts 154
LightSystem 155
Monitor 161
Monitor State Charts 172
Motor 175
Motor State Charts 176

Multi Window 177
Multi Window State Charts 187
Pool Spa 189
Pool Spa State Charts 193
Pre Amp Surround Sound Processor 195
Pre Amp Surround Sound Processor State Charts 204
Receiver 205
Receiver State Charts 219
Relay Device 221
RFIDSystem 223
Security System 229
Sensor Device 231
Sensor Device State Charts 232
Settop Box 233
Settop Box State Charts 243
Slide Projector 245
Slide Projector State Charts 247
Switcher 249
Text Keypad 255
Text Keypad State Charts 257
TV 259
TV State Charts 274
UPS 275
Utility 279
VCR 281
VCR State Charts 292
Video Conferencer 295
Video Conferencer State Charts 310

Video Processor 311
Video Processor State Charts 322
Video Projector 323
Video Projector State Charts 334
Video Wall 335
Video Wall State Charts 340
Volume Controller 341
Weather 343
Weather State Charts 347
Appendix - Deprecated Functions 349

Interfacing with Standard NetLinx API (SNAPI)

The Standard NetLinx API (SNAPI) maps function and feedback calls in Duet modules to ICSP channels, levels and commands. SNAPI allows NetLinx programmers to utilize Duet modules in their NetLinx programs and access the function and feedback of those modules through programming similar to programming they would use on an AMX device, such as a volume box. While each Duet module may support advanced functions via channels, levels and commands (see specific module documentation for the channels, levels and commands supported by that module), the SNAPI mappings apply to the Standard API supported by each module.

Overview

Devices

In Duet, all modules use a Duet virtual device. Duet virtual devices are in the range 41000:1:0 to 42000:1:0. Duet virtual devices are specifically designed for use with Duet modules. Regular virtual devices will not work properly with Duet modules.

Ports

In Duet, each device may support one or more ports. Multiple ports are used to provide access to different components within the module. For instance, a switcher may support output volume for each switcher output. In Duet, this is represented by a volume component for each output and in SNAPI, each of these components is mapped to a Duet virtual device ports. Port 1 will control the volume for output 1, port 2 will control the volume for output 2, etc… Another common use of port is to control different "zones" of and Audio Receiver, HVAC, Security Systems, etc… It is possible that a single port may offer only a small bit of functionality, such as volume control.

Port 1 is always the "main" device and supports all functions of the device. For components that have multiple instances, such as volume, port 1 will control the default component, which is usually component 1.

The documentation for each module will describe what ports are available and what functions they control. See the specific module documentation for a complete list of ports supported by the module.

Channels

In SNAPI, there are four kinds of channels: Input Function Channels, Momentary Function Channels, Discrete Function Channels and Feedback Channels.

Input function channels are used for response type functions, such as when a device wants to inform your program of an interesting event, similar to a button push on a touch panel. For instance, channel 1 is the input function channel for processButtonStateEvent (). When the module has information about the push or release of a button, the module will send a PUSH or RELEASE.

Momentary function channels are used for momentary type functions and do not provide discrete feedback. For instance, channel 9 is the momentary function channel for cyclePower(). When pulsed, the channel cycles the state of power on the device and only provides momentary feedback, i.e. the channel is on only while this function is activated.

Discrete function channels are used for discrete type functions and usually provide discrete feedback. For instance, channel 255 is the discrete function channel for setPower(); When turned on, this channel sets the state of the power on the device to on. When turned off, this channel sets the state of the power on the device to off. This channel provides discrete feedback as well; this channel is on if and only if the state of the power on the device is on. In most cases, the channel is listed as a Discrete function channel next to the function the channel controls and as a Feedback channel next to the feedback function that controls this channel.

Feedback channels provide discrete feedback only. For instance, channel 251 is the feedback channel of Communication Active. This channel is on if and only if the module is able to communicate to the device.

Levels

Levels in SNAPI are used for both function and feedback. In some cases, a level is only used for feedback, such as temperature, while some levels are used for function and feedback, such a volume. In most cases, level ranges are from 0-255. All exceptions to this rule are noted, and are only used when the level range is not bounded by a minimum and a maximum, such as temperature.

Commands

Commands in SNAPI are used for discrete and momentary functions when the function requires textual information, multiple parameters, or the functions are not commonly used. For instance, Temperature scale is set via a command because this is usually done only once in a control system program. Other functions, such as adding and removing lighting and keypad addresses, requires more information than a channel or level alone can convey. All commands start with a command header, followed by a "-" to separate the command from the data, and data arguments are usually separated by ","s.

Commands used to query for the status of a property start with a "?". Query commands cause the module to respond with a response command.

SNAPI.axi

SNAPI.axi is an include file that defines constants for each channel and level defined by SNAPI. These constants can be used in your programs in place of channel and level numbers. The constant names are listed in this document alongside every SNAPI function assigned to a channel or level.

SNAPI.axi is located in C:\Program Files\Common Files\AMXShare\AXIs. The file is organized by device type and lists all the standard channels and levels that may be supported by the module. This list does not contain all the channel or levels supported by a module and may include channels and levels not supported by a device. See the specific module documentation for a complete list of channels and levels supported by the module.

To include SNAPI.axi in your program, simply add an #INCLUDE statement for it:

```
#INCLUDE 'SNAPI.axi'
```

The file does not need to be copied to your project directory. The NetLinx compiler will be able to find this file automatically and include it in your program.

Programming

Channels

Input function channels are used for response type functions. For instance, channel 1 notifies your program of a button push in the KeypadComponent. When your program receives a PUSH, the button is pushed. When your program receives a RELEASE, the button is released. You should use BUTTON_EVENT's to capture the changes of an Input function channel:

```
BUTTON_EVENT[dvDevice, KEYPAD_BTN]
{
  PUSH:  // Button was pushed
  {
  }
  RELEASE: // Button was released
  {
  }
}
```

Momentary function channels are used to activate functions when the channels change from an OFF state to an ON state. For instance, channel 9 or the constant POWER cycles the state of the power on the device when it turns on. No change occurs when the channels change from an ON state to an OFF state. You should activate Momentary function channels using the PULSE, TO or MIN_TO keywords:

```
PULSE[dvDevice,POWER]// Cycle the state of power
TO[dvDevice,POWER]// Cycle the state of power
MIN_TO[dvDevice,POWER]// Cycle the state of power
```

Discrete function channels are used to activate functions when the channel changes from an OFF state to an ON state and from an ON to an OFF state. For instance, channel 255 or the constant POWER_ON sets the state of the power on the device when it turns on and off. You should activate discrete function channels using the ON and OFF keywords, or any syntax that changes the state of the channel such as a feedback assignment:

```
ON[dvDevice,POWER_ON]    // Turn the power on
OFF[dvDevice,POWER_ON]  // Turn the power off
[dvDevice,POWER_ON] = ![dvDevice,POWER_ON]  // Cycle the state of power
```

Feedback channels do not active function and should only be used for feedback. These channels can be used in CHANNEL_EVENTs or feedback assignment statements to read the status of the channel:

```
bCommunicationActive = [dvDevce,DEVICE_COMMUNICATING]
CHANNEL_EVENT[dvDevice,DEVICE_COMMUNICATING]
{
    ON:
        ON[bCommunicationActive]
    OFF:
        OFF[bCommunicationActive]
}
```

Ramping Channels

Some channels in SNAPI provide ramping functionality and some provide adjust "stepping" functionality. Since ramping on a device is only provided if the device supports ramping, a channel that causes ramping on one device may not cause ramping on another device. The following syntax can be used universally for all ramping functionality:

```
BUTTON_EVENT[dvTP,1]
{
    PUSH:
        TO[dvDevice,VOL_UP]
    HOLD[3 , REPEAT]:
        ON[dvDevce,VOL_UP]
}
```

The PUSH: TO part of the button event causes ramping to start and continue until the button is released. If the device does not support ramping, the device adjusts the desired parameter either up or down one step and stops. The HOLD: ON part of the button event causes the step adjustment to repeat, at a rate specified by the HOLD repeat time, until the button is released. The HOLD: ON part of the button event has no effect if the device supports ramping.

In a future version of Duet, it is expected that all modules will support ramping natively and that this NetLinx code will not always be required. However, if the module you are using does not support ramping, this code can be used to achieve ramping functionality.

Levels

Levels in SNAPI are used for both function and feedback. For feedback levels, the level value can be captured in a LEVEL_EVENT, with CREATE_LEVEL or sent directly to a touch panel display bargraph using DEFINE_CONNECT_LEVEL:

```
LEVEL_EVENT[dvDevice,1]
{
    // LEVEL.VALUE holds the new level value
}
CREATE_LEVEL dvDevice,1,nMyVariable // nMyVariable will hold the
                                    // latest value of the level
DEFINE_CONNECT_LEVEL(dvDevice,1,dvTp,1)
```

Levels used for functions can be set by calling SEND_LEVEL or by connecting to a touch panel active bargraph using DEFINE_CONNECT_LEVEL:

```
SEND_LEVEL dvDevice,1,nNewLevelValue
DEFINE_CONNECT_LEVEL(dvDevice,1,dvTp,1)
```

The CREATE_LEVEL/SEND_LEVEL mechanism is recommended for use with SNAPI. While LEVEL_EVENT will work fine, you may experience problems when a touch panel falls offline and then reconnects, which happens often with wireless panels. LEVEL_EVENT's will only fire when a change of the level value occurs. When the panel comes online, the only way to reliably update the level is with a SEND_LEVEL.

Commands

Commands in SNAPI are sent like commands to other devices, using the SEND_COMMAND keyword:

```
SEND_COMAND dvDevice,'?VERSION'
```

Commands used to query for the status of a property start with a "?". Query commands cause the module to respond with a response command. Note that this response is a command, not a string and can be captured in a DATA_EVENT in the COMMAND sub-section:

```
DATA_EVENT[dvDevice]
{
    COMMAND:
    {
        // DATA.TEXT holds the response to a query command
    }
}
```

General

The NetLinx program should assume that NetLinx levels are initially 0 and that channels are 'off'. The SNAPI router will notify the NetLinx client upon a change of state.

All Duet Virtual Devices should be created on port 1, e.g. 41000:1:0 in the following statements:

```
DEFINE_DEVICE
vdvModule = 41000:1:0
dvDevice =  135:1:0
DEFINE_MODULE 'LightModule' LightModule1 (vdvModule , dvDevice )
```

While it is possible to create a Duet Virtual Device on a port other than 1 and pass it to the Duet module, the behavior of the module is undefined.

Channel and Level Ranges

SNAPI uses only channels in range 1-299. Some channels are used for multiple functions but these channels belong to components that do not overlap within a single device. For instance, HVAC and Display both use channel 214 for setFanState and setFreezeOn respectively. Some channels are used for the same function in multiple components, for instance Video Conference and Display both define channel 191 for cyclePIPPosition. In both cases, this is by design.

Some devices may use custom channels for advanced functions. Channels 67-76 and 300-399 are reserved for modules to use for whatever functions they like. See specific module documentation for details on the channels used in that module.

SNAPI uses Levels in the range 1-48. Some levels are used for the same function in multiple components, for instance HVAC, Pool/Spa and Weather all define level 34 for Outdoor Temperature. This is by design.

Some device may use custom levels for advanced functions. Levels 50-80 and above are reserved for modules to use for whatever functions they like. See specific module documentation for details on the levels used in that module.

Commands and Escape Characters

SNAPI command uses comma as a parameter separator. If a parameter's value contains a comma, the parameter is escaping using double quotes at the start and end of the parameter. If a parameter's value contains a double quote character it is escaped with a pair of double quote characters.

The following examples are properly escaped parameter values:

- 6
- Hello
- Brown Eyed Girl
- "Morrison, Van"
- "Van ""The Man"" Morrison"

The following examples are improperly escaped parameter values:

- Morrison, Van
- Van "The Man" Morrison

SNAPI.axi includes a few helpful routines to build commands:

- DuetPackCmdHeader(Hdr)
- DuetPackCmdParam(Cmd, Param)
- DuetPackCmdParamArray(Cmd, Params[])

DuetPackCmdHeader is a command using a given command header where Hdr is the command header. DuetPackCmdParam adds a parameter to the command, escaping the parameter and adding parameter separators as needed; Cmd is the command to which the parameter is added and Param is the parameter to be added. DuetPackCmdParamArray is similar to DuetPackCmdParam but it takes an array of parameters and adds them to the command. All of these functions return the updated command.

SNAPI.axi includes a few helpful routines to parse commands as well:

- DuetParseCmdHeader(Cmd)
- DuetParseCmdParam(Cmd)

DuetParseCmdHeader removes and returns the command header from a command. DuetParseCmdParam removes and returns the next parameter from the command, un-escaping the parameter as needed. Both of these functions return a string containing the command header or the parameter.

An example program using these routines is shown below:

```
// Build a command to be stored in cTestCmd
cTestCmd = DuetPackCmdHeader('COMMAND')
cTestCmd = DuetPackCmdParam(cTestCmd,'Morrison,Van')
cTestCmd = DuetPackCmdParam(cTestCmd,'Wild Nights')
cTestCmd = DuetPackCmdParam(cTestCmd,'"The Man"')
cTestCmd = DuetPackCmdParam(cTestCmd,'Tupelo Honey')

// Resulting command is:
// 'COMMAND-"Morrison, Van",Wild Nights,""The Man"",Tupelo Honey'

// Remove the parameters for this command
cCmdheader = DuetParseCmdHeader(cTestCmd)
SWITCH (cCmdheader)
{
  CASE 'COMMAND':
  {
    cParam1 = DuetParseCmdParam(cTestCmd)
    cParam2 = DuetParseCmdParam(cTestCmd)
    cParam3 = DuetParseCmdParam(cTestCmd)
    cParam4 = DuetParseCmdParam(cTestCmd)

    // cParam1 = 'Morrison, Van'
    // cParam2 = 'Wild Nights'
    // cParam3 = '"The Man"'
    // cParam4 = 'Tupelo Honey'
  }
}
```

Amplifier

- IAmplifierComponent
- IAmplifierComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Amplifier						
Interface: IAmplifier						
Amplifier Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO

Amplifier Functions (Cont.)

Name	Component	Channel	Level	Command	Constant	Notes
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active

Amplifier Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Amplifier State Charts

Amplifier State Charts	
State	**Channel 255**
OFF	OFF
ON	ON

Standard NetLinx API (SNAPI): Devices

Audio Conferencer

- IAudioConferencerComponent
- IAudioConferencerComponentListener
- IDialerComponent
- IDialerComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPhonebookComponent
- IPhonebookComponentListener
- IPowerComponent
- IPowerComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Audio Conferencer						
Interface: IAudioConferencer						
Audio Conferencer Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cyclePrivacy()	IAudioConferencerComponent	145			ACONF_PRIVACY	Momentary Function Channel: Cycle privacy when channel is activated
setPrivacyOn(state)	IAudioConferencerComponent	146			ACONF_PRIVACY_ON	Discrete Function Channel: Privacy is on while channel is active
train()	IAudioConferencerComponent	147			ACONF_TRAIN	Momentary Function Channel: Train is executed when the channel is activated
processPrivacyEvent (boolean)	IAudioConferencerComponentListener	146			ACONF_PRIVACY_FB	Feedback Channel: Privacy is muted if channel is on
cycleAudibleRing()	IDialerComponent	205			DIAL_AUDIBLE_RING	Momentary Function Channel: Cycle audible ring when channel is activated
cycleAutoAnswer()	IDialerComponent	204			DIAL_AUTO_ANSWER	Momentary Function Channel: Cycle auto answer when channel is activated
cycleOffHook()	IDialerComponent	202			DIAL_OFF_HOOK	Momentary Function Channel: Cycle hook state when channel is activated

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
dial(recordID)	IDialerComponent			DIALID-<recordID>		Dial a speed dial record, where <id> is a record id returned by a speed dial search.
dial(index)	IDialerComponent			DIALINDEX-<index>		Dial a speed dial index, where <index> is 1 to x and x is the maximum supported speed dial index (see specific module documentation)
dialDTMF(char)	IDialerComponent			DTMF-<digit>		Send a DTMF tone for a character without regard for hook status
dialNumber(number)	IDialerComponent			DIALNUMBER-<number>		Dial a number where <number> is the number to be dialed.
flashHook()	IDialerComponent	208			DIAL_FLASH_HOOK	Momentary Function Channel: Flash hook when channel is activated
getDialerStatus()	IDialerComponent			?DIALERSTATUS		Querydialer status, responds with DIALER-STATUS-<status>, where <status> is DIALING,BUSY,RINGING,DISCONNECTED, NEGOTIATING,FAULT,CONNECTED
redial()	IDialerComponent	201			DIAL_REDIAL	Momentary Function Channel: Redial when channel is activated
setAudibleRingOn(state)	IDialerComponent	240			DIAL_AUDIBLE_RING_ON	Discrete Function Channel: Audible ring is on while channel is active
setAutoAnswerOn(state)	IDialerComponent	239			DIAL_AUTO_ANSWER_ON	Discrete Function Channel: Auto answer is on while channel is active
setOffHook(state)	IDialerComponent	238			DIAL_OFF_HOOK_ON	Discrete Function Channel: Hook state is off hook while channel is active
processAudibleRingOnEvent (boolean)	IDialerComponentListener	240			DIAL_AUDIBLE_RING_FB	Feedback Channel: Audible ring is on if channel is on
processAutoAnswerOnEvent (boolean)	IDialerComponentListener	239			DIAL_AUTO_ANSWER_FB	Feedback Channel: Auto answer is on if channel is on
processDialerStatusEvent (BUSY)	IDialerComponentListener			DIALERSTATUS-BUSY		Dialer status changed, number being dialed is busy
processDialerStatusEvent (CONNECTED)	IDialerComponentListener			DIALERSTATUS-CONNECTED		Dialer status changed, dialer is connected
processDialerStatusEvent (DIALING)	IDialerComponentListener			DIALERSTATUS-DIALING		Dialer status changed, dialer is dialing
processDialerStatusEvent (DISCONNECTED)	IDialerComponentListener			DIALERSTATUS-DISCONNECTED		Dialer status changed, dialer is disconnected/ idle
processDialerStatusEvent (FAULT)	IDialerComponentListener			DIALERSTATUS-FAULT		Dialer status changed, dialer encounter a fault during dialing/negotiating
processDialerStatusEvent (NEGOTIATING)	IDialerComponentListener			DIALERSTATUS-NEGOTIATING		Dialer status changed, dialer is negotiating

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDialerStatusEvent (RINGING)	IDialerComponentListener			DIALERSTATUS-RINGING		Dialer status changed, number being dialed is ringing
processIncomingCallEvent (String)	IDialerComponentListener			INCOMINGCALL-<number>		An incoming call is detected. If available via caller ID, the phone number will be supplied
processOffHookEvent (boolean)	IDialerComponentListener	238			DIAL_OFF_HOOK_FB	Feedback Channel: Hook state is off hook if channel is on
moveMenuCursor (UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor (DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor (LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor (RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor (UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor (UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor (DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor (DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton (DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton (DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton (DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton (DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton (DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton (DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton (DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton (DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton (DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton (PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton (ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton (CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton (MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton (EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton (VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton (THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton (THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton (ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton (REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton (LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton (SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton (PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton (FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton (SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton (XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton (FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton (CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton (BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton (FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton (ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton (DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton (HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton (LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton (LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton (RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton (UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton (DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton (ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton (DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton (POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton (COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton (DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton (CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton (PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton (DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton (SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton (INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton (FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton (CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton (RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton (GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton (PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton (PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton (DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton (TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton (PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton (AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton (HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton (TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton (ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton (ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton (AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton (PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton (SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton (SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton (FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton (RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton (INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties (fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent (byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
closeSearchDB (searchHandle)	IPhonebookComponent			PHONEBOOKCLOSESEARCH-<searchHandle>		Close the search associated with search key = <key>
deleteRecord (recordID)	IPhonebookComponent			PHONEBOOKDELETE-<recordID>		Delete phonebook record with index/id of <id>
getPhonebookCapacity()	IPhonebookComponent			?PHONEBOOKCAPACITY		Query for the phonebook capacity. Responds with PHONEBOOKCAPACITY-<count> where count is 1 to x and x is the maximum supported phonebook index (see specific module documentation)
queryDB(si)	IPhonebookComponent			PHONEBOOKNEXT-<searchHandle>[,<count>,<position>]		Request next <count> records starting with record <position> for the phonebook search associated with search key <key>. If <count> is not present, it is assumed to be 1. If <position> is not present, it is assumed to be the next record in the search result set based on the last PHONEBOOKNEXT or PHONEBOOKPREV command. Responses with PHONEBOOKNEXT-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
queryDB(si)	IPhonebookComponent			PHONEBOOKPREV-<searchHandle>[,<count>]		Request previous <count> records for the phonebook search associated with search key <key>. If <count> is not present, it is assumed to be 1. Responses with PHONEBOOKPREV-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
queryDB(si)	IPhonebookComponent			PHONEBOOKREFRESH-<searchHandle>[,<count>]		Refresh <count> records starting with the last starting position used with a PHONEBOOKNEXT or PHONEBOOKPREV command for the phonebook search associated with search key <key>. If <count> is not present, it is assumed to be 1. Responses with PHONEBOOKNEXT-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
searchDB(sr)	IPhonebookComponent			PHONEBOOKSEARCH-<searchHandle>,ID=<id>		Search the phonebook database for records with ID equal to <id>. If <id> is "*", all records are returned. <key> is a search key used in other search operations, such as closeSearchDB(). It can be any string you like, such as a panel device number or internal key that makes sense for your program. All future DB operation associated with this search will reference this key value.
updateRecord(sdr)	IPhonebookComponent			PHONEBOOKUPDATE-<recordID>,<name>,<number>		Update phonebook record with index/id of <id>, where <name> is the new name and <number> is the new number
processCloseEvent (String)	IPhonebookComponentListener			PHONEBOOKCLOSESEARCH-<searchHandle>		The search associated with search key <key> have been closed. No more operations can be performed against this search.
processDeleteRecordEvent (boolean,String)	IPhonebookComponentListener			PHONEBOOKDELETE-<recordID>,<success>		Phonebook record with index/id of <id> was deleted if <success> is 1, otherwise delete failed.
processEndOfSetEvent (String)	IPhonebookComponentListener			PHONEBOOKEND-<searchHandle>		End of search set was reached for search with search key <key>
processQueryDBEvent(PhonebookRecord)	IPhonebookComponentListener			PHONEBOOKRECORD-<searchHandle>,<recordID>,<resultNumber>,<name>,<number>		Phonebook record for search with search key <key>. <id> is the record ID, <#> is the record position from 1 to the total number of records, <name> is the name/label and <number> is the phone number.
processSearchDBEvent (PhonebookSearchSet)	IPhonebookComponentListener			PHONEBOOKSEARCHRESULT-<searchHandle>,<count>		Phonebook search results are available for search with search key <key>. <count> is the total count of records. Use PHONEBOOKNEXT and PHONEBOOKPREV to get more records.
processStartOfSetEvent(String)	IPhonebookComponentListener			PHONEBOOKSTART-<searchHandle>		Beginning of search set was reached for search with search key <key>
processUpdateRecordEvent (boolean,String)	IPhonebookComponentListener			PHONEBOOKUPDATE-<recordID>,<success>		Phonebook record with index/id of <id> was updated if <success> is 1, otherwise update failed.

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor (nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOL-PRESET-<preset>
saveVolumePreset (preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn (state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset (preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp (UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp (DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent (int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent (boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on

Audio Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processVolumePresetEvent (int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent (UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent (DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Audio Conferencer State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Audio Mixer

- IAudioMixerComponent
- IAudioMixerComponentListener
- IGainComponent
- IGainComponentListener
- IModuleComponent
- IModuleComponentListener
- ISwitcherComponent
- ISwitcherComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Audio Mixer						
Interface: IAudioMixer						
Audio Mixer Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getAudioMixerCrosspoint (input,output)	IAudioMixerComponent			?XPOINT-<input>,<output>		Query for Audio Mixer Crosspoint, responds with XPOINT-<value>,<input>,<output> where <value> is 0 to 255, <input> is 1 to the maximum supported input and <output> is the maximum supported output (see specific module documentation)
getAudioMixerPreset()	IAudioMixerComponent			?MIXERPRESET		Query for Audio Mixer Preset, responds with MIXERPRESET-<preset> where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
isAudioMixerCrosspointMuteOn (input,output)	IAudioMixerComponent			?XPOINTMUTE-<input>, <output>		Query for Audio Mixer Crosspoint Mute, responds with XPOINTMUTE-<state>,<input>,<output> where <state> is 0 (un-muted) or 1 (muted), <input> is 1 to the maximum supported input and <output> is the maximum supported output (see specific module documentation)
saveAudioMixerPreset (preset)	IAudioMixerComponent			MIXERPRESETSAVE-<preset>		Save Audio Mixer Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setAudioMixerCrosspoint (input,output[],value)	IAudioMixerComponent			XPOINT-<value>,<input>, <output,...>		Set Audio Mixer Crosspoint for <input> to one or more <output>s where <value> is 0 to 255. <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)

Audio Mixer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setAudioMixerCrosspointMuteOn (input,output,state)	IAudioMixerComponent			XPOINTMUTE-<state>,<input>, <output>		Set Audio Mixer Crosspoint Mute for <input> and <output> where <state> is 0 (un-muted) or 1 (muted). <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
setAudioMixerPreset (preset)	IAudioMixerComponent			MIXERPRESET-<preset>		Recall Audio Mixer Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
processAudioMixerCrosspointEvent (int,int,int)	IAudioMixerComponentListener			XPOINT-<value>,<input>, <output>		Audio Mixer Crosspoint changed for <input> to one or more <output>s where <value> is 0 to 255. <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
processAudioMixerCrosspointMuteOnEvent (int,int,boolean)	IAudioMixerComponentListener			XPOINTMUTE-<state>,<input>, <output>		Audio Mixer Crosspoint Mute changed for <input> and <output> where <state> is 0 (un-muted) or 1 (muted). <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
processAudioMixerPresetEvent (int)	IAudioMixerComponentListener			MIXERPRESET-<preset>		Mixer preset changed, where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
cycleGainMute()	IGainComponent	144			GAIN_MUTE	Momentary Function Channel: Cycle gain mute when channel is activated
setGain(level)	IGainComponent		5		GAIN_LVL	Set gain, range is 0-255
setGainMuteOn (state)	IGainComponent	143			GAIN_MUTE_ON	Discrete Function Channel: Gain mute is on while channel is active
setGainRamp (UP)	IGainComponent	140			GAIN_UP	Ramping Channel: Gain is ramped up while channel is active
setGainRamp (DOWN)	IGainComponent	141			GAIN_DN	Ramping Channel: Gain is ramped down while channel is active
processGainEvent (int)	IGainComponentListener		5		GAIN_LVL	Gain changed, range is 0-255
processGainMuteEvent (boolean)	IGainComponentListener	143			GAIN_MUTE_FB	Feedback Channel: Gain is muted if channel is on
processGainRampEvent (UP)	IGainComponentListener	140			GAIN_UP_FB	Feedback Channel: Gain is ramping up while channel is on
processGainRampEvent (DOWN)	IGainComponentListener	141			GAIN_DN_FB	Feedback Channel: Gain is ramping down while channel is on

Audio Mixer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

Audio Mixer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getInput(sl,output)	ISwitcherComponent			?INPUT-[<sl>,]<output>		Query for the input connected to an output, respond with SWITCH-L<sl>I<input>O <output> where <sl> is ALL, VIDEO, or AUDIO and <input> is 0 if there is no connection. If <sl> is not supplied, ALL will be assumed.
getOutput(sl,input)	ISwitcherComponent			?OUTPUT-[<sl>,]<input>		Query for the outputs connected to an input, respond with SWITCH-L<sl>I<input>O <output>,<output>... where <sl> is ALL, VIDEO, or AUDIO and <output> is 0 if there is no connection. If <sl> is not supplied, ALL will be assumed.
getSwitcherPreset()	ISwitcherComponent			?SWITCHPRESET		Query for switcher preset, responds with SWITCHPRESET-<preset>
saveSwitcherPreset(preset)	ISwitcherComponent			SWITCHPRESETSAVE-<preset>		Save switcher preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setSwitcherPreset(preset)	ISwitcherComponent			SWITCHPRESET-<preset>		Recall switcher preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
switchInputToOutput (AUDIO,input[],output[])	ISwitcherComponent			AI<input,...>O<output,...>		Switch <input> to one or more <output>s for switcher level Audio. Use <input> 0 for disconnect.
switchInputToOutput(AUDIO,input,output[])	ISwitcherComponent			AI<input>O<output,...>		Switch <input> to one or more <output>s for switcher level Audio. Use <input> 0 for disconnect.
switchInputToOutput(AUDIO,input,output)	ISwitcherComponent			AI<input>O<output>		Switch <input> to one or more <output>s for switcher level Audio. Use <input> 0 for disconnect.
switchInputToOutput(ALL,input[],output[])	ISwitcherComponent			CI<input,...>O<output,...>		Switch <input> to one or more <output>s for switcher level All. Use <input> 0 for disconnect.
switchInputToOutput(ALL,input,output[])	ISwitcherComponent			CI<input>O<output,...>		Switch <input> to one or more <output>s for switcher level All. Use <input> 0 for disconnect.
switchInputToOutput(ALL,input,output)	ISwitcherComponent			CI<input>O<output>		Switch <input> to one or more <output>s for switcher level All. Use <input> 0 for disconnect.
switchInputToOutput(sl,input,output[])	ISwitcherComponent			CL<sl>I<input>O<output,...>		Switch <input> to one or more <output>s where <sl> is ALL, VIDEO, or AUDIO. Use <input> 0 for disconnect.
switchInputToOutput(sl,input,output)	ISwitcherComponent			CL<sl>I<input>O<output>		Switch <input> to one or more <output>s where <sl> is ALL, VIDEO, or AUDIO. Use <input> 0 for disconnect.

Audio Mixer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
switchInputToOutput(VIDEO,input[],output[])	ISwitcherComponent			VI<input,...>O<output,...>		Switch <input> to one or more <output>s for switcher level Video. Use <input> 0 for dis-connect.
switchInputToOutput(VIDEO,input,output[])	ISwitcherComponent			VI<input>O<output,...>		Switch <input> to one or more <output>s for switcher level Video. Use <input> 0 for dis-connect.
switchInputToOutput(VIDEO,input,output)	ISwitcherComponent			VI<input>O<output>		Switch <input> to one or more <output>s where <sl> is ALL, VIDEO, or AUDIO. Use <input> 0 for disconnect.
processSwitcherPresetEvent(int)	ISwitcherComponentListener			SWITCHPRESET-<preset>		Switcher preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processSwitchEvent(SwitchLevel,int,int)	ISwitcherComponentListener			SWITCH-L<sl>I<input> O<output>		Switch connections changed, where <sl> is ALL, VIDEO, or AUDIO and <input> is 0 if there is no connection.
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on

Audio Mixer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Audio Processor

- IAudioProcessorComponent
- IAudioProcessorComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener

Name: Audio Processor						
Interface: IAudioProcessor						
Audio Processor Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
addAudioProcessorComponent (index,audioProcessorAddress)	IAudioProcessorComponent			AUDIOPROCADD-<index>, <audioProcessorAddress>		Add a Audio Processor object at a given index, where <index> is 1 through x and <address> is the object address and x is the maximum supported index (see specific module documentation)
adjustAudioProcessorLevel(1)	IAudioProcessorComponent	24			AUDIOPROC_LEVEL_UP	Ramping Channel: Audio Processor level is incremented when channel is activated
adjustAudioProcessorLevel(-1)	IAudioProcessorComponent	25			AUDIOPROC_LEVEL_DN	Ramping Channel: Audio Processor level is decremented when channel is activated
cycleAudioProcessorPreset()	IAudioProcessorComponent	209			AUDIOPROC_PRESET	Momentary Function Channel: Cycle Audio Processor preset when channel is activated
cycleAudioProcessorState()	IAudioProcessorComponent	26			AUDIOPROC_STATE	Momentary Function Channel: Cycle Audio Processor state when channel is activated
getAudioProcessorComponentAddress (index)	IAudioProcessorComponent			?AUDIOPROCADDR-<index>		Query for the address of the Audio Processor object at index <index>, responds with AUDIOPROCADDR-<index>,<address>
getAudioProcessorComponentIndex (audioProcessorAddress)	IAudioProcessorComponent			?AUDIOPROCIDX-<address>		Query for the index of the Audio Processor object with address <address>, responds with AUDIOPROCADDR-<index>,<address>
getAudioProcessorCrosspoint (input,output)	IAudioProcessorComponent			?XPOINT-<input>,<output>		Query for Audio Processor crosspoint, responds with XPOINT-<value>,<input>,<output> where <value> is 0 to 255, <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
getAudioProcessorInput (output)	IAudioProcessorComponent			?INPUT-<output>		Query for the input connected to an output, respond with SWITCH-L<sl>I<input>O<output> where <sl> is AUDIO and <input> is 0 if there is no connection.

Audio Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getAudioProcessorOutput(input)	IAudioProcessorComponent			?OUTPUT-<input>		Query for the outputs connected to an input, respond with SWITCH-L<sl>I<input>O <output>,<output>... where <sl> is AUDIO and <input> is 0 if there is no connection.
getAudioProcessorPreset()	IAudioProcessorComponent			?AUDIOPROCPRESET		Query for Audio Processor Preset, responds with AUDIOPROCPRESET-<preset> where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
isAudioProcessorCrosspointMuteOn (input,output)	IAudioProcessorComponent			?XPOINTMUTE-<input>, <output>		Query for Audio Processor Crosspoint Mute, responds with XPOINTMUTE-<state>,<input>,<output> where <state> is 0 (un-muted) or 1 (muted), <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
removeAudioProcessorComponent (audioProcessorAddress)	IAudioProcessorComponent			AUDIOPROCREMOVEADDR-<audioProcessorAddress>		Remove the Audio Processor object with address <address>, where <address> is the object address
removeAudioProcessorComponent (index)	IAudioProcessorComponent			AUDIOPROCREMOVEIDX-<index>		Remove the Audio Processor object at index <index>, where <index> is 1 through x and x is the maximum supported index (see specific module documentation)
saveAudioProcessorPreset (preset)	IAudioProcessorComponent			AUDIOPROCPRESETSAVE-<preset>		Save Audio Processor Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setAudioProcessorCrosspoint (input,output[],value)	IAudioProcessorComponent			XPOINT-<value>,<input>, <output>, <output>, <output...>		Set Audio Processor crosspoint for <input> to one or more <output>s where <value> is 0 to 255. <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
setAudioProcessorCrosspointMuteOn (input,output,state)	IAudioProcessorComponent			XPOINTMUTE-<state>, <input>,<output>		Set Audio Processor Crosspoint Mute for <input> and <output> where <state> is 0 (un-muted) or 1 (muted). <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
setAudioProcessorLevel(level)	IAudioProcessorComponent		1		AUDIOPROC_LVL	Set Audio Processor level, range is 0-255
setAudioProcessorPreset(preset)	IAudioProcessorComponent			AUDIOPROCPRESET-<preset>		Recall Audio Processor Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)

Audio Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setAudioProcessorStateOn(state)	IAudioProcessorComponent	199			AUDIOPROC_STATE_ON	Discrete Function Channel: Audio Processor state is on while channel is active
switchAudioProcessorInputToOutput (input,output[])	IAudioProcessorComponent			AI<input>O<output>, <output,...>		Switch <input> to one or more <output>s for switcher level Audio. Use <input> 0 for disconnect.
processAudioProcessorCrosspointEvent (int,int,int)	IAudioProcessorComponentListener			XPOINT-<value>,<input>, <output>		Audio Processor crosspoint changed for <input> to one or more <output>s where <value> is 0 to 255. <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
processAudioProcessorCrosspoint MuteOnEvent(int,int,boolean)	IAudioProcessorComponentListener			XPOINTMUTE-<state>, <input>,<output>		Audio Processor Crosspoint Mute changed for <input> and <output> where <state> is 0 (un-muted) or 1 (muted). <input> is 1 to the maximum supported input and <output> is 1 to the maximum supported output (see specific module documentation)
processAudioProcessorLevelEvent (int)	IAudioProcessorComponentListener		1		AUDIOPROC_LVL	Audio Processor level changed, range is 0-255
processAudioProcessorPresetEvent (int)	IAudioProcessorComponentListener			AUDIOPROCPRESET-<preset>		Audio Processor preset changed, where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
processAudioProcessorStateOnEvent (boolean)	IAudioProcessorComponentListener	199			AUDIOPROC_STATE_FB	Feedback Channel: Audio Processor state is on if channel is on
processAudioProcessorSwitchEvent (int,int)	IAudioProcessorComponentListener			SWITCH-L<sl>I<input> O<output>		Audio Processor switch connections changed, where <sl> is AUDIO and <input> is 0 if there is no connection.
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device

Audio Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Audio Processor State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Audio Tape

- IAudioTapeComponent
- IAudioTapeComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ITapeTransportComponent
- ITapeTransportComponentListener

Name: Audio Tape						
Interface: IAudioTape						
Audio Tape Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleAudioTapeRecordMute()	IAudioTapeComponent	128			CASS_RECORD_MUTE	Momentary Function Channel: Cycle record mute when channel is activated
cycleAudioTapeSide()	IAudioTapeComponent	42			CASS_TAPE_SIDE	Momentary Function Channel: Cycle tape side when channel is activated
reversePlay()	IAudioTapeComponent	41			CASS_REVERSE_PLAY	Momentary Function Channel: Reverse direction of play (but not the side). The audio will be played backwards.
setAudioTapeRecordMuteOn(state)	IAudioTapeComponent	200			CASS_RECORD_MUTE_ON	Discrete Function Channel: Record Mute is on while channel is active
setAudioTapeSide(A)	IAudioTapeComponent	126			CASS_TAPE_SIDE_A	Momentary Function Channel: Set tape side to side A
setAudioTapeSide(B)	IAudioTapeComponent	127			CASS_TAPE_SIDE_B	Momentary Function Channel: Set tape side to side B
processAudioTapeRecordMuteOnEvent (boolean)	IAudioTapeComponentListener	200			CASS_RECORD_MUTE_FB	Feedback Channel: Record mute is on while channel is active
processAudioTapeSideEvent(A)	IAudioTapeComponentListener	126			CASS_TAPE_SIDE_A_FB	Feedback Channel: Tape side is set to side A
processAudioTapeSideEvent(B)	IAudioTapeComponentListener	127			CASS_TAPE_SIDE_B_FB	Feedback Channel: Tape side is set to side B
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>

Audio Tape Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied file-name. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Audio Tape Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleSearchSpeed()	ITapeTransportComponent	119			SEARCH_SPEED	Momentary Function Channel: Cycle search speed when channel is activated
ejectTape()	ITapeTransportComponent	120			EJECT	Momentary Function Channel: Tape is ejected when channel is activated
resetTapeCounter()	ITapeTransportComponent	121			RESET_COUNTER	Momentary Function Channel: Counter is reset when channel is activated
setTapeCounterNotificationOn(state)	ITapeTransportComponent			TAPECOUNTERNOTIFY-<state>		Turn counter notification on or off, where <state> is 1 or 0
setTapeTransport(PLAY)	ITapeTransportComponent	1			PLAY	Momentary Function Channel: Deck is set to play when the channel is activated
setTapeTransport(STOP)	ITapeTransportComponent	2			STOP	Momentary Function Channel: Deck is set to stop when the channel is activated
setTapeTransport(PAUSE)	ITapeTransportComponent	3			PAUSE	Momentary Function Channel: Deck is set to pause when the channel is activated
setTapeTransport(FF)	ITapeTransportComponent	4			FFWD	Momentary Function Channel: Deck is set to fast-forward when the channel is activated
setTapeTransport(REW)	ITapeTransportComponent	5			REW	Momentary Function Channel: Deck is set to rewind when the channel is activated
setTapeTransport(SEARCH_FWD)	ITapeTransportComponent	6			SFWD	Momentary Function Channel: Deck is set to search forward when the channel is activated
setTapeTransport(SEARCH_REV)	ITapeTransportComponent	7			SREV	Momentary Function Channel: Deck is set to search reverse when the channel is activated
setTapeTransport(RECORD)	ITapeTransportComponent	8			RECORD	Momentary Function Channel: Deck is set to record when the channel is activated
setTapeTransport(SLOW_FWD)	ITapeTransportComponent	188			SLOW_FWD	Momentary Function Channel: Deck is set to slow forward when the channel is activated
setTapeTransport(SLOW_REV)	ITapeTransportComponent	189			SLOW_REV	Momentary Function Channel: Deck is set to slow reverse when the channel is activated
processTapeCounterEvent(MediaTime)	ITapeTransportComponentListener			TAPECOUNTER-<counter>		Tape counter changed, <counter> is hh:mm:ss.ff
processTapeLoadedEvent(boolean)	ITapeTransportComponentListener	122			TAPE_LOADED_FB	Feedback Channel: Tape is loaded while channel is on
processTapeRecordLockedEvent (boolean)	ITapeTransportComponentListener	123			RECORD_LOCK_FB	Feedback Channel: Tape is record locked while channel is on

Audio Tape Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processTapeTransportEvent(PLAY)	ITapeTransportComponentListener	241			PLAY_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent(STOP)	ITapeTransportComponentListener	242			STOP_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent(PAUSE)	ITapeTransportComponentListener	243			PAUSE_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent(FF)	ITapeTransportComponentListener	244			FFWD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent(REW)	ITapeTransportComponentListener	245			REW_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SEARCH_FWD)	ITapeTransportComponentListener	246			SFWD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SEARCH_REV)	ITapeTransportComponentListener	247			SREV_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent(RECORD)	ITapeTransportComponentListener	248			RECORD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SLOW_FWD)	ITapeTransportComponentListener	249			SLOW_FWD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SLOW_REV)	ITapeTransportComponentListener	250			SLOW_REV_FB	Feedback Channel: Transport state change (see chart below)

Audio Tape State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

processTapeTransportEvent										
State	**Channel 241**	**Channel 242**	**Channel 243**	**Channel 244**	**Channel 245**	**Channel 246**	**Channel 247**	**Channel 248**	**Channel 249**	**Channel 250**
PLAY	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
STOP	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
FF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
REW	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
SEARCH_FWD	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SEARCH_REV	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
RECORD	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
RECORD_PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	ON	OFF	OFF
SLOW_FWD	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SLOW_REV	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

Audio Tuner Device

- IAudioTunerDeviceComponent
- IAudioTunerDeviceComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ITunerStationComponent
- ITunerStationComponentListener

Name: Audio Tuner Device						
Interface: IAudioTunerDevice						
Audio Tuner Device Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>

Audio Tuner Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleBand()	ITunerStationComponent	40			TUNER_BAND	Momentary Function Channel: Cycle tuner band when channel is activated
cycleDisplayInfo()	ITunerStationComponent	234			TUNER_OSD	Momentary Function Channel: Cycle on-screen or front-panel display info when channel is activated
cycleStationPresetGroup()	ITunerStationComponent	224			TUNER_PRESET_GROUP	Momentary Function Channel: Cycle station preset group/bank when channel is activated
decrementStation()	ITunerStationComponent	226			TUNER_STATION_DN	Momentary Function Channel: Station is decremented when channel is activated
getBand()	ITunerStationComponent			?BAND		Query for band, responds with BAND-<tb> where <tb> is AM,FM, FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV

Audio Tuner Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getStation()	ITunerStationComponent			?XCH		Query for station, responds with XCH-<station> where <station> is a station string such as "501", "103.7" or "5.1"
getStationPreset()	ITunerStationComponent			?TUNERPRESET		Query for tuner preset, responds with TUNERPRESET-<preset>
getStationPresetCount()	ITunerStationComponent			?STATIONPRESETCOUNT		Query for number of valid station presets, responds with STATIONPRESETCOUNT-<count>
getStationPresetProperties()	ITunerStationComponent			?STATIONPRESET PROPERTIES		Query input properties for all station presets, responds with STATIONPRESETPROPERTIES-<index>,<displayName>,<station>[;<index>,<displayName>,<station>]*
getStationPresetProperty(index)	ITunerStationComponent			?STATIONPRESET PROPERTY-<index>		Query properties for a single station preset, responds with STATIONPRESETPROPERTY-<index>,<displayName>,<station>
getStationPresetSelect()	ITunerStationComponent			?STATIONPRESET SELECT		Query for the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
getTunerBandProperties()	ITunerStationComponent			?TUNERBAND PROPERTIES		Query input properties for all tuner bands, responds with TUNERBANDPROPERTIES-<index>,<displayName>,<tb>[,<index>,<displayName>,<tb>]*
getTunerBandProperty(index)	ITunerStationComponent			?TUNERBAND PROPERTY-<index>		Query properties for a single tuner band, responds with TUNERBANDPROPERTY-<index>,<displayName>,<tb>
getTunerBandPropertyCount()	ITunerStationComponent			?TUNERBAND PROPERTYCOUNT		Query for number of valid tuner bands, responds with TUNERBANDPROPERTY-COUNT-<count>
getTunerBandSelect()	ITunerStationComponent			?TUNERBANDSELECT		Query for the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
getTunerComponentProperties()	ITunerStationComponent			?TUNERCOMPONENT PROPERTIES		Query properties for a single tuner component, responds with TUNERCOMPONENTPROPERTIES-<index>,<displayName>,<value>[;<index>,<displayName>,<value>]*
getTunerComponentProperty()	ITunerStationComponent			?TUNERCOMPONENT PROPERTY		Query input properties for all inputs, responds with TUNERCOMPONENTPROPERTY-<index>,<displayName>,<value>

Audio Tuner Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getTunerComponentPropertyCount()	ITunerStationComponent			?TUNERCOMPONENT PROPERTYCOUNT		Query for number of tuner components, responds with TUNERCOMPONENTPROPERTYCOUNT-<count>
getTunerComponentSelect()	ITunerStationComponent			?TUNERCOMPONENT SELECT		Query for the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
gotoPreviousStation()	ITunerStationComponent	235			TUNER_PREV	Momentary Function Channel: Previous station is selected when channel is activated
incrementStation()	ITunerStationComponent	225			TUNER_STATION_UP	Momentary Function Channel: Station is incremented when channel is activated
nextStationPreset()	ITunerStationComponent	22			CHAN_UP	Momentary Function Channel: Next station preset is selected when channel is activated
previousStationPreset()	ITunerStationComponent	23			CHAN_DN	Momentary Function Channel: Previous station preset is selected when channel is activated
scanStation(FORWARD)	ITunerStationComponent	227			TUNER_SCAN_FWD	Momentary Function Channel: Scans for next station while channel is activate.
scanStation(BACKWARD)	ITunerStationComponent	228			TUNER_SCAN_REV	Momentary Function Channel: Scans for previous station while channel is activate
seekStation(FORWARD)	ITunerStationComponent	229			TUNER_SEEK_FWD	Momentary Function Channel: Seeks for next station while channel is activate
seekStation(BACKWARD)	ITunerStationComponent	230			TUNER_SEEK_REV	Momentary Function Channel: Seeks for previous station while channel is activate
setBand(tb)	ITunerStationComponent			BAND-<tb>		Set band, where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV
setStation(station)	ITunerStationComponent			XCH-<station>		Set station, where <station> is a station string such as "501", "103.7" or "5.1"
setStationPreset(preset)	ITunerStationComponent			TUNERPRESET-<preset>		Recall tuner preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setStationPresetSelect(index)	ITunerStationComponent			STATIONPRESET SELECT-<index>		Set the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
setTunerBandSelect(index)	ITunerStationComponent			TUNERBAND SELECT-<index>		Set the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>

Audio Tuner Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setTunerComponentSelect(index)	ITunerStationComponent			TUNERCOMPONENT SELECT-<index>		Set the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
processBandEvent(TunerBand)	ITunerStationComponentListener			BAND-<band>		Band changed, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV
processStationEvent(String)	ITunerStationComponentListener			XCH-<station>		Station changed, where <station> is a station string such as "501", "103.7" or "5.1"
processStationPresetCountEvent(count)	ITunerStationComponentListener			STATIONPRESET COUNT-<count>		Response to ?TUNERCOUNT, where <count> is the integer number of tuner components
processStationPresetEvent(int)	ITunerStationComponentListener			TUNERPRESET-<int>		Tuner preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processStationPresetPropertiesEvent (String[])	ITunerStationComponentListener			STATIONPRESET PROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>,<value>;...]		Returns the properties for each station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetPropertyEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetSelectEvent(index)	ITunerStationComponentListener			STATIONPRESET SELECT-<index>		Returns the selected station preset index
processTunerBandPropertiesEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTIES-<index>, <displayName>, <tunerBand>[;<index>, <displayName>, <tunerBand>;...]		Returns properties for each tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandPropertyCountEvent (count)	ITunerStationComponentListener			TUNERBANDPROPERTY COUNT-<count>		Returns the number of valid tuner bands
processTunerBandPropertyEvent(String[])	ITunerStationComponentListener			TUNERBANDPROPERTY-<index>,<displayName>, <tunerBand>		Returns properties for a single tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandSelectEvent(index)	ITunerStationComponentListener			TUNERBAND SELECT-<index>		Returns the selected tuner band index

Audio Tuner Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processTunerComponentPropertiesEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTIES-<index>, <displayName>,<value> [;<index>,<displayName>, <value>;...]		Returns the properties for each tuner component
processTunerComponentPropertyCountEvent (count)	ITunerStationComponentListener			TUNERCOMPONENT PROPERTYCOUNT		Returns the integer number of tuner components
processTunerComponentPropertyEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single tuner component
processTunerComponentSelectEvent(index)	ITunerStationComponentListener			TUNERCOMPONENT SELECT-<index>		Returns the selected tuner component index

Audio Tuner Device State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Camera

- ICameraComponent
- ICameraComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener

Name: Camera

Interface: ICamera

Camera Functions

Name	Component	Channel	Level	Command	Constant	Notes
cycleAutoFocus()	ICameraComponent	172			AUTO_FOCUS	Momentary Function Channel: Cycle auto focus when channel is activated
cycleAutoIris()	ICameraComponent	173			AUTO_IRIS	Momentary Function Channel: Cycle auto iris when channel is activated
cycleCameraPreset()	ICameraComponent	177			CAM_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getCameraComponentCount()	ICameraComponent			?CAMERACOMPONENT COUNT		Query number of camera components, responds with CAMERACOMPONENTCOUNT-<count>
getCameraPreset()	ICameraComponent			?CAMERAPRESET		Query for camera preset, responds with CAMERAPRESET-<preset>
getCameraPresetCount()	ICameraComponent			?CAMERAPRESET COUNT		Query number of presets on a camera, responds with CAMERAPRESETCOUNT-<count>
getCameraPresetProperties()	ICameraComponent			?CAMERAPRESET PROPERTIES		Query properties for every camera preset, responds with CAMERAPRESETPROPERTIES-<index>, <displayName>[;<index>,<displayName>]
getCameraPresetProperty(index)	ICameraComponent			?CAMERAPRESET PROPERTY		Query properties for a single camera preset, responds with CAMERAPRESETPROPERTY-<index>, <displayName>
saveCameraPreset(preset)	ICameraComponent			CAMERAPRESETSAVE-<preset>		Save Camera Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setAutoFocusOn(state)	ICameraComponent	162			AUTO_FOCUS_ON	Discrete Function Channel: Auto focus is on while channel is active
setAutoIrisOn(state)	ICameraComponent	163			AUTO_IRIS_ON	Discrete Function Channel: Auto iris is on while channel is active
setCameraPreset(preset)	ICameraComponent			CAMERAPRESET-<preset>		Recall camera preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setFocus(focus)	ICameraComponent		16		FOCUS_LVL	Set focus position, range is 0-255 (0=near)

Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setFocusRamp(NEAR)	ICameraComponent	160			FOCUS_NEAR	Ramping Channel: Focus is ramped near while channel is active
setFocusRamp(FAR)	ICameraComponent	161			FOCUS_FAR	Ramping Channel: Focus is ramped far while channel is active
setFocusSpeed(speed)	ICameraComponent		19		FOCUS_SPEED_LVL	Set focus speed, range is 0-255 (0=slow)
setIris(iris)	ICameraComponent		17		IRIS_LVL	Set iris position, range is 0-255 (0=closed)
setIrisRamp(OPEN)	ICameraComponent	174			IRIS_OPEN	Ramping Channel: Iris is ramped open while channel is active
setIrisRamp(CLOSE)	ICameraComponent	175			IRIS_CLOSE	Ramping Channel: Iris is ramped closed while channel is active
setIrisSpeed(speed)	ICameraComponent		20		IRIS_SPEED_LVL	Set iris speed, range is 0-255 (0=slow)
setPan(pan)	ICameraComponent		27		PAN_LVL	Set pan position, range is 0-255 (0=left)
setPanRamp(LEFT)	ICameraComponent	134			PAN_LT	Ramping Channel: Pan is ramped left while channel is active
setPanRamp(RIGHT)	ICameraComponent	135			PAN_RT	Ramping Channel: Pan is ramped right while channel is active
setPanSpeed(speed)	ICameraComponent		29		PAN_SPEED_LVL	Set pan speed, range is 0-255 (0=slow)
setTilt(tilt)	ICameraComponent		28		TILT_LVL	Set tilt position, range is 0-255 (0=down)
setTiltRamp(UP)	ICameraComponent	132			TILT_UP	Ramping Channel: Tilt is ramped up while channel is active
setTiltRamp(DOWN)	ICameraComponent	133			TILT_DN	Ramping Channel: Tilt is ramped down while channel is active
setTiltSpeed(speed)	ICameraComponent		30		TILT_SPEED_LVL	Set tilt speed, range is 0-255 (0=slow)
setZoom(zoom)	ICameraComponent		15		ZOOM_LVL	Set zoom position, range is 0-255 (0=out/Wide)
setZoomRamp(OUT)	ICameraComponent	158			ZOOM_OUT	Ramping Channel: Zoom is ramped out (wide) while channel is active
setZoomRamp(IN)	ICameraComponent	159			ZOOM_IN	Ramping Channel: Zoom is ramped in (tele) while channel is active
setZoomSpeed(speed)	ICameraComponent		18		ZOOM_SPEED_LVL	Set zoom speed, range is 0-255 (0=slow)
processAutoFocusEvent(boolean)	ICameraComponentListener	162			AUTO_FOCUS_FB	Feedback Channel: Auto focus is on if channel is active
processAutoIrisEvent(boolean)	ICameraComponentListener	163			AUTO_IRIS_FB	Feedback Channel: Auto iris is on if channel is active
processCameraComponentCountEvent (count)	ICameraComponentListener			CAMERACOMPONENT COUNT-<count>		Returns integer number of camera components
processCameraPresetCountEvent (count)	ICameraComponentListener			CAMERAPRESETCOUNT-<count>		Returns integer number of camera presets

Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processCameraPresetEvent(int)	ICameraComponentListener			CAMERAPRESET-<preset>		Camera preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processCameraPresetPropertiesEvent (String[])	ICameraComponentListener			CAMERAPRESET PROPERTIES-<index>, <displayName>[;-<index>, <displayName>,]*		Returns string containing <index> and <displayName> for each camera preset.
processCameraPresetPropertyEvent (String[])	ICameraComponentListener			CAMERAPRESET PROPERTY-<index>, <displayName>		Returns string containing <index> and <displayName> for a camera preset.
processFocusEvent(int)	ICameraComponentListener		16		FOCUS_LVL	Focus changed, range is 0-255 (0=near)
processFocusRampEvent(NEAR)	ICameraComponentListener	160			FOCUS_NEAR_FB	Feedback Channel: Focus is ramping near while channel is on
processFocusRampEvent(FAR)	ICameraComponentListener	161			FOCUS_FAR_FB	Feedback Channel: Focus is ramping far while channel is on
processFocusSpeedEvent(int)	ICameraComponentListener		19		FOCUS_SPEED_LVL	Focus speed changed, range is 0-255 (0=slow)
processIrisEvent(int)	ICameraComponentListener		17		IRIS_LVL	Iris changed, range is 0-255 (0=closed)
processIrisRampEvent(OPEN)	ICameraComponentListener	174			IRIS_OPEN_FB	Feedback Channel: Iris is ramping open while channel is on
processIrisRampEvent(CLOSE)	ICameraComponentListener	175			IRIS_CLOSE_FB	Feedback Channel: Iris is ramping closed while channel is on
processIrisSpeedEvent(int)	ICameraComponentListener		20		IRIS_SPEED_LVL	Iris speed changed, range is 0-255 (0=slow)
processPanEvent(int)	ICameraComponentListener		27		PAN_LVL	Pan changed, range is 0-255 (0=left)
processPanRampEvent(LEFT)	ICameraComponentListener	134			PAN_LT_FB	Feedback Channel: Pan is ramping left while channel is on
processPanRampEvent(RIGHT)	ICameraComponentListener	135			PAN_RT_FB	Feedback Channel: Pan is ramping right up while channel is on
processPanSpeedEvent(int)	ICameraComponentListener		29		PAN_SPEED_LVL	Pan speed changed, range is 0-255 (0=slow)
processTiltEvent(int)	ICameraComponentListener		28		TILT_LVL	Tilt changed, range is 0-255 (0=down)
processTiltRampEvent(UP)	ICameraComponentListener	132			TILT_UP_FB	Feedback Channel: Tilt is ramping up while channel is on
processTiltRampEvent(DOWN)	ICameraComponentListener	133			TILT_DN_FB	Feedback Channel: Tilt is ramping down while channel is on
processTiltSpeedEvent(int)	ICameraComponentListener		30		TILT_SPEED_LVL	Tilt speed changed, range is 0-255 (0=slow)
processZoomEvent(int)	ICameraComponentListener		15		ZOOM_LVL	Zoom changed, range is 0-255 (0=out/wide)
processZoomRampEvent(OUT)	ICameraComponentListener	158			ZOOM_OUT_FB	Feedback Channel: Zoom is ramping out (wide) while channel is on

Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processZoomRampEvent(IN)	ICameraComponentListener	159			ZOOM_IN_FB	Feedback Channel: Zoom is ramping in (tele) while channel is on
processZoomSpeedEvent(int)	ICameraComponentListener		18		ZOOM_SPEED_LVL	Zoom speed changed, range is 0-255 (0=slow)
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated

Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Camera State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Digital Media Decoder

- IDigitalMediaDecoderComponent
- IDigitalMediaDecoderComponentListener
- IDiscTransportComponent
- IDiscTransportComponentListener
- IMediaDeviceComponent
- IMediaDeviceComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Digital Media Decoder						
Interface: IDigitalMediaDecoder						
Digital Media Decoder Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleScanSpeed()	IDiscTransportComponent	192			SCAN_SPEED	Momentary Function Channel: Cycles the scan speed when the channel is activated
getTrackInfo()	IDiscTransportComponent			?TRACKINFO		Query for the track info, responds with TRACKINFO-<num>,<duration>, <discNumber>, where <duration> is a String in the format hh:mm:ss.ff
queryTrackProperties()	IDiscTransportComponent			?TRACKPROPS		Query for the track properties, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
queryTrackProperty(key)	IDiscTransportComponent			?TRACKPROP-<keyName>		Query for a track property, responds with TRACKPROP-<key>,<value> command
setDiscTransport(PLAY)	IDiscTransportComponent	1			PLAY	Momentary Function Channel: Deck is set to play when the channel is activated
setDiscTransport(STOP)	IDiscTransportComponent	2			STOP	Momentary Function Channel: Deck is set to stop when the channel is activated
setDiscTransport(PAUSE)	IDiscTransportComponent	3			PAUSE	Momentary Function Channel: Deck is set to pause when the channel is activated
setDiscTransport(NEXT)	IDiscTransportComponent	4			FFWD	Momentary Function Channel: Deck is set to next track/chapter when the channel is activated

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setDiscTransport(PREVIOUS)	IDiscTransportComponent	5			REW	Momentary Function Channel: Deck is set to previous track/chapter when the channel is activated
setDiscTransport(SCAN_FWD)	IDiscTransportComponent	6			SFWD	Momentary Function Channel: Deck is set to scan forward when the channel is activated
setDiscTransport(SCAN_REV)	IDiscTransportComponent	7			SREV	Momentary Function Channel: Deck is set to scan reverse when the channel is activated
setDiscTransport(RECORD)	IDiscTransportComponent	8			RECORD	Momentary Function Channel: Deck is set to record when the channel is activated
setDiscTransport(FRAME_FWD)	IDiscTransportComponent	185			FRAME_FWD	Momentary Function Channel: Deck is set to step frame forward when the channel is activated
setDiscTransport(FRAME_REV)	IDiscTransportComponent	186			FRAME_REV	Momentary Function Channel: Deck is set to step frame reverse when the channel is activated
setDiscTransport(SLOW_FWD)	IDiscTransportComponent	188			SLOW_FWD	Momentary Function Channel: Deck is set to slow forward when the channel is activated
setDiscTransport(SLOW_REV)	IDiscTransportComponent	189			SLOW_REV	Momentary Function Channel: Deck is set to slow reverse when the channel is activated
setPlayPosition(mt)	IDiscTransportComponent			PLAYPOSITION-<counter>		Set the play position where <counter> is a String in the format [-]hh:mm:ss.ff, mm should be 0 >= mm < 60, ss should be 0 >= ss < 60, ff should be valid for the disc type
setTrackCounterNotificationOn(state)	IDiscTransportComponent			TRACKCOUNTERNOTIFY-<state>		Turn track counter notification on or off, where <state> is 1 or 0
processDiscTransportEvent(PLAY)	IDiscTransportComponentListener	241			PLAY_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(STOP)	IDiscTransportComponentListener	242			STOP_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(PAUSE)	IDiscTransportComponentListener	243			PAUSE_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SCAN_FWD)	IDiscTransportComponentListener	246			SFWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SCAN_REV)	IDiscTransportComponentListener	247			SREV_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(RECORD)	IDiscTransportComponentListener	248			RECORD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SLOW_FWD)	IDiscTransportComponentListener	249			SLOW_FWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SLOW_REV)	IDiscTransportComponentListener	250			SLOW_REV_FB	Feedback Channel: Transport state change (see chart below)

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processTrackCounterEvent(MediaTime)	IDiscTransportComponentListener			TRACKCOUNTER-<counter>		Track counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processTrackInfoEvent(TrackInfo)	IDiscTransportComponentListener			TRACKINFO-<num>,<duration>,<discNumber>		Track info changed, where <num> is track number, <duration> is a track duration String in the format [-]hh:mm:ss.ff and <discNumber> is the disc number the track belongs to.
processTrackPropertiesEvent (Properties)	IDiscTransportComponentListener			TRACKPROP-<key>,<value>		Track properties query response, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
cycleMediaDeviceRandom()	IMediaDeviceComponent	124			MEDIA_RANDOM	Momentary Function Channel: Cycle random when channel is activated
cycleMediaDeviceRepeat()	IMediaDeviceComponent	125			MEDIA_REPEAT	Momentary Function Channel: Cycle repeat when channel is activated
getMediaDeviceSource()	IMediaDeviceComponent			?DECODESOURCE		Query for Media Device Source, responds with DECODESOURCE-<url> where <url> is the URL of the source.
queryMediaDeviceProperties()	IMediaDeviceComponent			?DECODEPROPS		Query for all Media Device Properties, responds with multiple DECODEPROP-<key>,<value>, one for each property, where <key> is the property key and <value> is the property value.
queryMediaDeviceProperty(sKeyName)	IMediaDeviceComponent			?DECODEPROP-<keyName>		Query for one Media Device Properties, responds with DECODEPROP-<key>,<value> where <key> is the property key and <value> is the property value.
setMediaDeviceCounterNotificationOn (state)	IMediaDeviceComponent			MEDIACOUNTERNOTIFY-<state>		Turn media counter notification on or off, where <state> is 1 or 0
setMediaDeviceRandomState (RANDOM_DISC)	IMediaDeviceComponent	178			MEDIA_RANDOM_DISC_ON	Momentary Function Channel: Random-disc is on while channel is active
setMediaDeviceRandomState (RANDOM_ALL)	IMediaDeviceComponent	179			MEDIA_RANDOM_ALL_ON	Momentary Function Channel: Random-all is on while channel is active
setMediaDeviceRandomState (RANDOM_OFF)	IMediaDeviceComponent	180			MEDIA_RANDOM_OFF_ON	Momentary Function Channel: Random-off is on while channel is active
setMediaDeviceRepeatState (REPEAT_DISC)	IMediaDeviceComponent	181			MEDIA_REPEAT_DISC_ON	Momentary Function Channel: Repeat-disc is on while channel is active
setMediaDeviceRepeatState (REPEAT_TRACK)	IMediaDeviceComponent	182			MEDIA_REPEAT_TRACK_ON	Momentary Function Channel: Repeat-track is on while channel is active
setMediaDeviceRepeatState (REPEAT_ALL)	IMediaDeviceComponent	183			MEDIA_REPEAT_ALL_ON	Momentary Function Channel: Repeat-all is on while channel is active
setMediaDeviceRepeatState (REPEAT_OFF)	IMediaDeviceComponent	184			MEDIA_REPEAT_OFF_ON	Momentary Function Channel: Repeat-off is on while channel is active

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setMediaDeviceSource(cURISource)	IMediaDeviceComponent			DECODESOURCE-<url>		Set Media Device Source where <url> is the URL of the source.
processMediaDeviceCounterEvent (MediaTime)	IMediaDeviceComponentListener			MEDIACOUNTER-<counter>		Media counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processMediaDevicePropertiesEvent (Properties)	IMediaDeviceComponentListener			DECODEPROP-<key>, <value>		Media property value where <key> is the property key and <value> is the property value. One command is returned for each key.
processMediaDeviceRandomStateEvent (RANDOM_DISC)	IMediaDeviceComponentListener	178			MEDIA_RANDOM_DISC_FB	Feedback Channel: Random state change (see chart below)
processMediaDeviceRandomStateEvent (RANDOM_ALL)	IMediaDeviceComponentListener	179			MEDIA_RANDOM_ALL_FB	Feedback Channel: Random state change (see chart below)
processMediaDeviceRandomStateEvent (RANDOM_OFF)	IMediaDeviceComponentListener	180			MEDIA_RANDOM_OFF_FB	Feedback Channel: Random state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_DISC)	IMediaDeviceComponentListener	181			MEDIA_REPEAT_DISC_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_TRACK)	IMediaDeviceComponentListener	182			MEDIA_REPEAT_TRACK_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_ALL)	IMediaDeviceComponentListener	183			MEDIA_REPEAT_ALL_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_OFF)	IMediaDeviceComponentListener	184			MEDIA_REPEAT_OFF_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceSourceInfoEvent (MediaDeviceSourceInfo)	IMediaDeviceComponentListener			DECODESOURCE-<recordID>,<name>, <url>		Media Device Source changed where <recordID> is the record ID of the source (may be blank), <name> is the name of the source and <url> is the URL of the source.
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated

Digital Media Decoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Digital Media Decoder State Charts

processDiscTransportEvent								
State	**Channel 241**	**Channel 242**	**Channel 243**	**Channel 246**	**Channel 247**	**Channel 248**	**Channel 249**	**Channel 250**
PLAY	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
STOP	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
SCAN_FWD	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SCAN_REV	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
RECORD	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
RECORD_PAUSE	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF
SLOW_FWD	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SLOW_REV	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

processMediaDeviceRandomStateEvent			
State	**Channel 178**	**Channel 179**	**Channel 180**
RANDOM_DISC	ON	OFF	OFF
RANDOM_ALL	OFF	ON	OFF
RANDOM_OFF	OFF	OFF	ON

processMediaDeviceRepeatStateEvent				
State	**Channel 181**	**Channel 182**	**Channel 183**	**Channel 184**
REPEAT_DISC	ON	OFF	OFF	OFF
REPEAT_TRACK	OFF	ON	OFF	OFF
REPEAT_ALL	OFF	OFF	ON	OFF
REPEAT_OFF	OFF	OFF	OFF	ON

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Digital Media Encoder

- IDigitalMediaEncoderComponent
- IDigitalMediaEncoderComponentListener
- IDiscTransportComponent
- IDiscTransportComponentListener
- IGainComponent
- IGainComponentListener
- IModuleComponent
- IModuleComponentListener
- IOutputStreamComponent
- IOutputStreamComponentListener
- IPowerComponent
- IPowerComponentListener

Name: Digital Media Encoder						
Interface: IDigitalMediaEncoder						
Digital Media Encoder Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleScanSpeed()	IDiscTransportComponent	192			SCAN_SPEED	Momentary Function Channel: Cycles the scan speed when the channel is activated
getTrackInfo()	IDiscTransportComponent			?TRACKINFO		Query for the track info, responds with TRACKINFO-<num>,<duration>, <discNumber>, where <duration> is a String in the format hh:mm:ss.ff
queryTrackProperties()	IDiscTransportComponent			?TRACKPROPS		Query for the track properties, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
queryTrackProperty(key)	IDiscTransportComponent			?TRACKPROP-<keyName>		Query for a track property, responds with TRACKPROP-<key>,<value> command
setDiscTransport(PLAY)	IDiscTransportComponent	1			PLAY	Momentary Function Channel: Deck is set to play when the channel is activated
setDiscTransport(STOP)	IDiscTransportComponent	2			STOP	Momentary Function Channel: Deck is set to stop when the channel is activated
setDiscTransport(PAUSE)	IDiscTransportComponent	3			PAUSE	Momentary Function Channel: Deck is set to pause when the channel is activated
setDiscTransport(NEXT)	IDiscTransportComponent	4			FFWD	Momentary Function Channel: Deck is set to next track/chapter when the channel is activated
setDiscTransport(PREVIOUS)	IDiscTransportComponent	5			REW	Momentary Function Channel: Deck is set to previous track/chapter when the channel is activated

Digital Media Encoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setDiscTransport(SCAN_FWD)	IDiscTransportComponent	6			SFWD	Momentary Function Channel: Deck is set to scan forward when the channel is activated
setDiscTransport(SCAN_REV)	IDiscTransportComponent	7			SREV	Momentary Function Channel: Deck is set to scan reverse when the channel is activated
setDiscTransport(RECORD)	IDiscTransportComponent	8			RECORD	Momentary Function Channel: Deck is set to record when the channel is activated
setDiscTransport(FRAME_FWD)	IDiscTransportComponent	185			FRAME_FWD	Momentary Function Channel: Deck is set to step frame forward when the channel is activated
setDiscTransport(FRAME_REV)	IDiscTransportComponent	186			FRAME_REV	Momentary Function Channel: Deck is set to step frame reverse when the channel is activated
setDiscTransport(SLOW_FWD)	IDiscTransportComponent	188			SLOW_FWD	Momentary Function Channel: Deck is set to slow forward when the channel is activated
setDiscTransport(SLOW_REV)	IDiscTransportComponent	189			SLOW_REV	Momentary Function Channel: Deck is set to slow reverse when the channel is activated
setPlayPosition(mt)	IDiscTransportComponent			PLAYPOSITION-<counter>		Set the play position where <counter> is a String in the format [-]hh:mm:ss.ff, mm should be 0 >= mm < 60, ss should be 0 >= ss < 60, ff should be valid for the disc type
setTrackCounterNotificationOn(state)	IDiscTransportComponent			TRACKCOUNTERNOTIFY-<state>		Turn track counter notification on or off, where <state> is 1 or 0
processDiscTransportEvent(PLAY)	IDiscTransportComponentListener	241			PLAY_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(STOP)	IDiscTransportComponentListener	242			STOP_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(PAUSE)	IDiscTransportComponentListener	243			PAUSE_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(SCAN_FWD)	IDiscTransportComponentListener	246			SFWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(SCAN_REV)	IDiscTransportComponentListener	247			SREV_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(RECORD)	IDiscTransportComponentListener	248			RECORD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(SLOW_FWD)	IDiscTransportComponentListener	249			SLOW_FWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(SLOW_REV)	IDiscTransportComponentListener	250			SLOW_REV_FB	Feedback Channel: Transport state change (see chart below)

Digital Media Encoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processTrackCounterEvent(MediaTime)	IDiscTransportComponentListener			TRACKCOUNTER-<counter>		Track counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processTrackInfoEvent(TrackInfo)	IDiscTransportComponentListener			TRACKINFO-<num>, <duration>,<discNumber>		Track info changed, where <num> is track number, <duration> is a track duration String in the format [-]hh:mm:ss.ff and <discNumber> is the disc number the track belongs to.
processTrackPropertiesEvent (Properties)	IDiscTransportComponentListener			TRACKPROP-<key>, <value>		Track properties query response, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
cycleGainMute()	IGainComponent	144			GAIN_MUTE	Momentary Function Channel: Cycle gain mute when channel is activated
setGain(level)	IGainComponent		5		GAIN_LVL	Set gain, range is 0-255
setGainMuteOn(state)	IGainComponent	143			GAIN_MUTE_ON	Discrete Function Channel: Gain mute is on while channel is active
setGainRamp(UP)	IGainComponent	140			GAIN_UP	Ramping Channel: Gain is ramped up while channel is active
setGainRamp(DOWN)	IGainComponent	141			GAIN_DN	Ramping Channel: Gain is ramped down while channel is active
processGainEvent(int)	IGainComponentListener		5		GAIN_LVL	Gain changed, range is 0-255
processGainMuteEvent(boolean)	IGainComponentListener	143			GAIN_MUTE_FB	Feedback Channel: Gain is muted if channel is on
processGainRampEvent(UP)	IGainComponentListener	140			GAIN_UP_FB	Feedback Channel: Gain is ramping up while channel is on
processGainRampEvent(DOWN)	IGainComponentListener	141			GAIN_DN_FB	Feedback Channel: Gain is ramping down while channel is on
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device

Digital Media Encoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
addOutputStreamSink(cURISink)	IOutputStreamComponent			ENCODESINKADD-<url>		Add an Output Stream Sink where <url> is the URL of the sink to be added.
getOutputStreamSinks()	IOutputStreamComponent			?ENCODESINK		Query for Output Stream Sinks, responds with multiple ENCODESINK-<url>, one for each output stream sink, where <url> is the URL of the sink/destination.
getOutputStreamSource()	IOutputStreamComponent			?ENCODESOURCE		Query for Output Stream Source, responds with ENCODESOURCE-<url> where <url> is the URL for the source.
queryOutputStreamProperties()	IOutputStreamComponent			?ENCODEPROPS		Query for all Output Stream Media Properties, responds with multiple ENCODEPROP-<key>,<value>, one for each property, where <key> is the property key and <value> is the property value.
queryOutputStreamProperty(sName)	IOutputStreamComponent			?ENCODEPROP-<keyName>		Query for one Output Stream Media Properties, responds with ENCODEPROP-<key>,<value> where <key> is the property key and <value> is the property value.
removeOutputStreamSink(cURISink)	IOutputStreamComponent			ENCODESINKREMOVE-<url>		Remove an Output Stream Sink where <url> is the URL of the sink to be removed.
setOutputStreamProperty(sName,sValue)	IOutputStreamComponent			ENCODEPROP-<keyName>,<value>		Set an Output Stream Property where <key> is the property key and <value> is the property value.

Digital Media Encoder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setOutputStreamSource(cURISource)	IOutputStreamComponent			ENCODESOURCE-<url>		Set Output Stream Source where <url> is the URL for the source.
processOutputStreamPropertiesEvent (Properties)	IOutputStreamComponentListener			ENCODEPROP-<key>, <value>		Output Stream Property value where <key> is the property key and <value> is the property value. One command is returned for each key.
processOutputStreamSinkAddEvent(URL)	IOutputStreamComponentListener			ENCODESINKADD-<url>		Output Stream Sink added where <url> is the URL of the added sink.
processOutputStreamSinkRemoveEvent (URL)	IOutputStreamComponentListener			ENCODESINKREMOVE-<url>		Output Stream Sink removed where <url> is the URL of the removed sink.
processOutputStreamSourceEvent(URL)	IOutputStreamComponentListener			ENCODESOURCE-<url>		Output Stream Source changed where <url> is the URL for the source.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Digital Media Encoder State Charts

processDiscTransportEvent								
State	**Channel 241**	**Channel 242**	**Channel 243**	**Channel 246**	**Channel 247**	**Channel 248**	**Channel 249**	**Channel 250**
PLAY	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
STOP	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
SCAN_FWD	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SCAN_REV	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
RECORD	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
RECORD_PAUSE	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF
SLOW_FWD	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SLOW_REV	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Digital Media Server

- IDigitalMediaServerComponent
- IDigitalMediaServerComponentListener
- IDiscTransportComponent
- IDiscTransportComponentListener
- IGainComponent
- IGainComponentListener
- IMediaDBComponent
- IMediaDBComponentListener
- IMediaDeviceComponent
- IMediaDeviceComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IOutputStreamComponent
- IOutputStreamComponentListener
- IPowerComponent
- IPowerComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Digital Media Server						
Interface: IDigitalMediaServer						
Digital Media Server Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleScanSpeed()	IDiscTransportComponent	192			SCAN_SPEED	Momentary Function Channel: Cycles the scan speed when the channel is activated
getTrackInfo()	IDiscTransportComponent			?TRACKINFO		Query for the track info, responds with TRACKINFO-<num>,<duration>, <discNumber>, where <duration> is a String in the format hh:mm:ss.ff
queryTrackProperties()	IDiscTransportComponent			?TRACKPROPS		Query for the track properties, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
queryTrackProperty(key)	IDiscTransportComponent			?TRACKPROP-<keyName>		Query for a track property, responds with TRACKPROP-<key>,<value> command
setDiscTransport(PLAY)	IDiscTransportComponent	1			PLAY	Momentary Function Channel: Deck is set to play when the channel is activated
setDiscTransport(STOP)	IDiscTransportComponent	2			STOP	Momentary Function Channel: Deck is set to stop when the channel is activated

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setDiscTransport(PAUSE)	IDiscTransportComponent	3			PAUSE	Momentary Function Channel: Deck is set to pause when the channel is activated
setDiscTransport(NEXT)	IDiscTransportComponent	4			FFWD	Momentary Function Channel: Deck is set to next track/chapter when the channel is activated
setDiscTransport(PREVIOUS)	IDiscTransportComponent	5			REW	Momentary Function Channel: Deck is set to previous track/chapter when the channel is activated
setDiscTransport(SCAN_FWD)	IDiscTransportComponent	6			SFWD	Momentary Function Channel: Deck is set to scan forward when the channel is activated
setDiscTransport(SCAN_REV)	IDiscTransportComponent	7			SREV	Momentary Function Channel: Deck is set to scan reverse when the channel is activated
setDiscTransport(RECORD)	IDiscTransportComponent	8			RECORD	Momentary Function Channel: Deck is set to record when the channel is activated
setDiscTransport(FRAME_FWD)	IDiscTransportComponent	185			FRAME_FWD	Momentary Function Channel: Deck is set to step frame forward when the channel is activated
setDiscTransport(FRAME_REV)	IDiscTransportComponent	186			FRAME_REV	Momentary Function Channel: Deck is set to step frame reverse when the channel is activated
setDiscTransport(SLOW_FWD)	IDiscTransportComponent	188			SLOW_FWD	Momentary Function Channel: Deck is set to slow forward when the channel is activated
setDiscTransport(SLOW_REV)	IDiscTransportComponent	189			SLOW_REV	Momentary Function Channel: Deck is set to slow reverse when the channel is activated
setPlayPosition(mt)	IDiscTransportComponent			PLAYPOSITION-<counter>		Set the play position where <counter> is a String in the format [-]hh:mm:ss.ff, mm should be 0 >= mm < 60, ss should be 0 >= ss < 60, ff should be valid for the disc type
setTrackCounterNotificationOn(state)	IDiscTransportComponent			TRACKCOUNTERNOTIFY-<state>		Turn track counter notification on or off, where <state> is 1 or 0
processDiscTransportEvent(PLAY)	IDiscTransportComponentListener	241			PLAY_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(STOP)	IDiscTransportComponentListener	242			STOP_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(PAUSE)	IDiscTransportComponentListener	243			PAUSE_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SCAN_FWD)	IDiscTransportComponentListener	246			SFWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SCAN_REV)	IDiscTransportComponentListener	247			SREV_FB	Feedback Channel: Transport state change (see chart below)

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDiscTransportEvent(RECORD)	IDiscTransportComponentListener	248			RECORD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SLOW_FWD)	IDiscTransportComponentListener	249			SLOW_FWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SLOW_REV)	IDiscTransportComponentListener	250			SLOW_REV_FB	Feedback Channel: Transport state change (see chart below)
processTrackCounterEvent (MediaTime)	IDiscTransportComponentListener			TRACKCOUNTER-<counter>		Track counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processTrackInfoEvent(TrackInfo)	IDiscTransportComponentListener			TRACKINFO-<num>,<duration>, <discNumber>		Track info changed, where <num> is track number, <duration> is a track duration String in the format [-]hh:mm:ss.ff and <discNum-ber> is the disc number the track belongs to.
processTrackPropertiesEvent (Properties)	IDiscTransportComponentListener			TRACKPROP-<key>, <value>		Track properties query response, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
cycleGainMute()	IGainComponent	144			GAIN_MUTE	Momentary Function Channel: Cycle gain mute when channel is activated
setGain(level)	IGainComponent		5		GAIN_LVL	Set gain, range is 0-255
setGainMuteOn(state)	IGainComponent	143			GAIN_MUTE_ON	Discrete Function Channel: Gain mute is on while channel is active
setGainRamp(UP)	IGainComponent	140			GAIN_UP	Ramping Channel: Gain is ramped up while channel is active
setGainRamp(DOWN)	IGainComponent	141			GAIN_DN	Ramping Channel: Gain is ramped down while channel is active
processGainEvent(int)	IGainComponentListener		5		GAIN_LVL	Gain changed, range is 0-255
processGainMuteEvent(boolean)	IGainComponentListener	143			GAIN_MUTE_FB	Feedback Channel: Gain is muted if channel is on
processGainRampEvent(UP)	IGainComponentListener	140			GAIN_UP_FB	Feedback Channel: Gain is ramping up while channel is on
processGainRampEvent(DOWN)	IGainComponentListener	141			GAIN_DN_FB	Feedback Channel: Gain is ramping down while channel is on
closeSearchDB(searchHandle)	IMediaDBComponent			MEDIADBCLOSESEARCH-<searchHandle>		Close the search associated with search key = <key>
deleteRecord(recordID)	IMediaDBComponent			MEDIADBDELETE-<recordID>		Delete media record with record id of <id>

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
queryDB(mdbss)	IMediaDBComponent			MEDIADBNEXT-<searchHandle>		Request next record (count =1) starting with the next record <position> for the media search associated with search key <key>. If <position> is not present, it is assumed to be the next record in the search result set based on the last MEDIADBNEXT or MEDIADBPREV command. Responds with MEDIADBNEXT-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
queryDB(mdbss,count)	IMediaDBComponent			MEDIADBNEXT-<searchHandle>,<count>		Request next record (count =1) starting with record <position> for the media search associated with search key <key>. If <position> is not present, it is assumed to be the next record in the search result set based on the last MEDIADBNEXT or MEDIADBPREV command. Responses with MEDIADBNEXT-<key>,<count>,<position> where <key> is the search
queryDB(mdbss)	IMediaDBComponent			MEDIADBPREV-<searchHandle>		Request previous record (count = 1) for the media search associated with search key <key>. Responses with MEDIADBPREV-<key>,1,<position> where <key> is the search key and <position> is the position of the first record to be returned between 1 through the total number of records.
queryDB(mdbss,count)	IMediaDBComponent			MEDIADBPREV-<searchHandle>,<count>		Request previous <count> records for the media search associated with search key <key>. Responses with MEDIADBPREV-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
queryDB(mdbss)	IMediaDBComponent			MEDIADBREFRESH-<searchHandle>		Refresh record starting with the last starting position used with a MEDIADBNEXT or MEDIADBPREV command for the media search associated with search key <key>. Responses with MEDIADBNEXT-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
queryDB(mdbss,count)	IMediaDBComponent			MEDIADBREFRESH-<searchHandle>,<count>		Refresh record starting with the last starting position used with a MEDIADBNEXT or MEDIADBPREV command for the media search associated with search key <key>. Responses with MEDIADBNEXT-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
queryMediaDBProperties(recordID)	IMediaDBComponent			?MEDIADBPROPS-<recordID>		Query for all Media Database Properties for a given record where <id> is the record ID, responds with multiple MEDIADBPROP-<id>,<key>,<value>, one for each property, where <id> is the record id of the record from which to retrieve the properties, <key> is the property key and <value> is the property value.
queryMediaDBProperty(id,key)	IMediaDBComponent			?MEDIADBPROP-<recordID>,<keyName>		Query for one Media Database Property where <id> is the record ID and <key> is the property key to query, responds with MEDIADBPROP-<id>,<key>,<value> where <id> is the record id of the record from which to retrieve the properties, <key> is the property key and <value> is the property value.
searchDB(sr)	IMediaDBComponent			MEDIADBSEARCH-<searchHandle>,<search type>=<search string>,RETURN=<return type>		Search the media database for records with <search type> equal to <search string>. If <search string> is "*", all records are returned. <key> is a search key used in other search operations, such as closeSearchDB(). It can be any string you like, such as a panel device number or internal key that makes sense for your program. All future DB operation associated with this search will reference this key value. <search type> can be ALL, ID, ARTIST, GENRE, TITLE, KEYWORDS, PLAYLIST, BOOKMARK. RETURN= is optional and limits the type of items returned in the result set where <result type> can be ALL, PICTURE, APPLICATION, TRACK, CHAPTER, PLAYLIST, BOOKMARK, DISC, AUDIO, VIDEO, GENRE, ARTIST, STATION.

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setMediaDBProperty(sID,sName,sValue)	IMediaDBComponent			MEDIADBPROP-<recordID>,<keyName>,<value>		Set a Media Database Property where <id> is the record id of the record for which to set the properties, <key> is the property key and <value> is the property value.
updateRecord(rec)	IMediaDBComponent			MEDIADBUPDATE-<recordID>,<name>,<record type>[,<url>]		Update media record with record id of <id>, where <name> is the new name and <record type> is the new record type, i.e. PICTURE, APPLICATION, TRACK, CHAPTER, PLAYLIST, BOOKMARK, DISC, AUDIO, VIDEO, GENRE, ARTIST, STATION and <url> is the existing media URL to associate with the new/updated record
processCloseEvent(String)	IMediaDBComponentListener			MEDIADBCLOSESEARCH-<searchHandle>		The search associated with search key <key> have been closed. No more operations can be performed against this search.
processDeleteRecordEvent (String, boolean)	IMediaDBComponentListener			MEDIADBDELETE-<recordID>,<success>		Media record with record id of <id> was deleted if <success> is 1, otherwise delete failed.
processEndOfSetEvent(String)	IMediaDBComponentListener			MEDIADBEND-<searchHandle>		End of search set was reached for search with search key <key>
processMediaDBPropertiesEvent (String, Properties)	IMediaDBComponentListener			MEDIADBPROP-<recordID>,<keyName>,<value>		Media Database property value where <id> is the record ID, <key> is the property key and <value> is the property value. One command is returned for each key.
processQueryDBEvent(MediaRecord)	IMediaDBComponentListener			MEDIADBRECORD-<key>,<recordID>,<resultNumber>,<name>,<record type>,<url>		Media record for search with search key <key>. <id> is the record ID, <#> is the record position from 1 to the total number of records, <name> is the item name, <record type> is the record type, which could be PICTURE, APPLICATION, TRACK, CHAPTER, PLAYLIST, BOOKMARK, DISC, AUDIO, VIDEO, GENRE, ARTIST, STATION and <url> is the URL of the media.
processSearchDBEvent (MediaSearchSet)	IMediaDBComponentListener			MEDIADBSEARCHRESULT-<searchHandle>,<count>		Media search results are available for search with search key <key>. <count> is the total count of records. Use MEDIADBNEXT and MEDIADBPREV to get more records.
processStartOfSetEvent(String)	IMediaDBComponentListener			MEDIADBSTART-<searchHandle>		Beginning of search set was reached for search with search key <key>
processUpdateRecordEvent (String, boolean)	IMediaDBComponentListener			MEDIADBUPDATE-<recordID>,<success>		Media record with record id of <id> was updated if <success> is 1, otherwise update failed.

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleMediaDeviceRandom()	IMediaDeviceComponent	124			MEDIA_RANDOM	Momentary Function Channel: Cycle random when channel is activated
cycleMediaDeviceRepeat()	IMediaDeviceComponent	125			MEDIA_REPEAT	Momentary Function Channel: Cycle repeat when channel is activated
getMediaDeviceSource()	IMediaDeviceComponent			?DECODESOURCE		Query for Media Device Source, responds with DECODESOURCE-<url> where <url> is the URL of the source.
queryMediaDeviceProperties()	IMediaDeviceComponent			?DECODEPROPS		Query for all Media Device Properties, responds with multiple DECODEPROP-<key>,<value>, one for each property, where <key> is the property key and <value> is the property value.
queryMediaDeviceProperty(sKeyName)	IMediaDeviceComponent			?DECODEPROP-<keyName>		Query for one Media Device Properties, responds with DECODEPROP-<key>,<value> where <key> is the property key and <value> is the property value.
setMediaDeviceCounterNotificationOn (state)	IMediaDeviceComponent			MEDIACOUNTERNOTIFY-<state>		Turn media counter notification on or off, where <state> is 1 or 0
setMediaDeviceRandomState (RANDOM_DISC)	IMediaDeviceComponent	178			MEDIA_RANDOM_DISC_ON	Momentary Function Channel: Random-disc is on while channel is active
setMediaDeviceRandomState (RANDOM_ALL)	IMediaDeviceComponent	179			MEDIA_RANDOM_ALL_ON	Momentary Function Channel: Random-all is on while channel is active
setMediaDeviceRandomState (RANDOM_OFF)	IMediaDeviceComponent	180			MEDIA_RANDOM_OFF_ON	Momentary Function Channel: Random-off is on while channel is active
setMediaDeviceRepeatState (REPEAT_DISC)	IMediaDeviceComponent	181			MEDIA_REPEAT_DISC_ON	Momentary Function Channel: Repeat-disc is on while channel is active
setMediaDeviceRepeatState (REPEAT_TRACK)	IMediaDeviceComponent	182			MEDIA_REPEAT_TRACK_ON	Momentary Function Channel: Repeat-track is on while channel is active
setMediaDeviceRepeatState (REPEAT_ALL)	IMediaDeviceComponent	183			MEDIA_REPEAT_ALL_ON	Momentary Function Channel: Repeat-all is on while channel is active
setMediaDeviceRepeatState (REPEAT_OFF)	IMediaDeviceComponent	184			MEDIA_REPEAT_OFF_ON	Momentary Function Channel: Repeat-off is on while channel is active
setMediaDeviceSource(cURISource)	IMediaDeviceComponent			DECODESOURCE-<url>		Set Media Device Source where <url> is the URL of the source.
processMediaDeviceCounterEvent (MediaTime)	IMediaDeviceComponentListener			MEDIACOUNTER-<counter>		Media counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processMediaDevicePropertiesEvent (Properties)	IMediaDeviceComponentListener			DECODEPROP-<key>, <value>		Media property value where <key> is the property key and <value> is the property value. One command is returned for each key.

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processMediaDeviceRandomStateEvent(RANDOM_DISC)	IMediaDeviceComponentListener	178			MEDIA_RANDOM_DISC_FB	Feedback Channel: Random state change (see chart below)
processMediaDeviceRandomStateEvent (RANDOM_ALL)	IMediaDeviceComponentListener	179			MEDIA_RANDOM_ALL_FB	Feedback Channel: Random state change (see chart below)
processMediaDeviceRandomStateEvent (RANDOM_OFF)	IMediaDeviceComponentListener	180			MEDIA_RANDOM_OFF_FB	Feedback Channel: Random state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_DISC)	IMediaDeviceComponentListener	181			MEDIA_REPEAT_DISC_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_TRACK)	IMediaDeviceComponentListener	182			MEDIA_REPEAT_TRACK_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_ALL)	IMediaDeviceComponentListener	183			MEDIA_REPEAT_ALL_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceRepeatStateEvent (REPEAT_OFF)	IMediaDeviceComponentListener	184			MEDIA_REPEAT_OFF_FB	Feedback Channel: Repeat state change (see chart below)
processMediaDeviceSourceInfoEvent (MediaDeviceSourceInfo)	IMediaDeviceComponentListener			DECODESOURCE-<recordID>,<name>,<url>		Media Device Source changed where <recordID> is the record ID of the source (may be blank), <name> is the name of the source and <url> is the URL of the source.
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
addOutputStreamSink(cURISink)	IOutputStreamComponent			ENCODESINKADD-<url>		Add an Output Stream Sink where <url> is the URL of the sink to be added.
getOutputStreamSinks()	IOutputStreamComponent			?ENCODESINK		Query for Output Stream Sinks, responds with multiple ENCODESINK-<url>, one for each output stream sink, where <url> is the URL of the sink/destination.
getOutputStreamSource()	IOutputStreamComponent			?ENCODESOURCE		Query for Output Stream Source, responds with ENCODESOURCE-<url> where <url> is the URL for the source.
queryOutputStreamProperties()	IOutputStreamComponent			?ENCODEPROPS		Query for all Output Stream Media Properties, responds with multiple ENCODEPROP-<key>,<value>, one for each property, where <key> is the property key and <value> is the property value.
queryOutputStreamProperty(sName)	IOutputStreamComponent			?ENCODEPROP-<keyName>		Query for one Output Stream Media Properties, responds with ENCODEPROP-<key>,<value> where <key> is the property key and <value> is the property value.
removeOutputStreamSink(cURISink)	IOutputStreamComponent			ENCODESINKREMOVE-<url>		Remove an Output Stream Sink where <url> is the URL of the sink to be removed.
setOutputStreamProperty (sName,sValue)	IOutputStreamComponent			ENCODEPROP-<keyName>,<value>		Set an Output Stream Property where <key> is the property key and <value> is the property value.
setOutputStreamSource(cURISource)	IOutputStreamComponent			ENCODESOURCE-<url>		Set Output Stream Source where <url> is the URL for the source.
processOutputStreamPropertiesEvent (Properties)	IOutputStreamComponentListener			ENCODEPROP-<key>, <value>		Output Stream Property value where <key> is the property key and <value> is the property value. One command is returned for each key.

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processOutputStreamSinkAddEvent (URL)	IOutputStreamComponentListener			ENCODESINKADD-<url>		Output Stream Sink added where <url> is the URL of the added sink.
processOutputStreamSinkRemoveEvent (URL)	IOutputStreamComponentListener			ENCODESINKREMOVE-<url>		Output Stream Sink removed where <url> is the URL of the removed sink.
processOutputStreamSourceEvent(URL)	IOutputStreamComponentListener			ENCODESOURCE-<url>		Output Stream Source changed where <url> is the URL for the source.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active

Digital Media Server Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Digital Media Server State Charts

processDiscTransportEvent								
State	**Channel 241**	**Channel 242**	**Channel 243**	**Channel 246**	**Channel 247**	**Channel 248**	**Channel 249**	**Channel 250**
PLAY	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
STOP	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
SCAN_FWD	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SCAN_REV	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
RECORD	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
RECORD_PAUSE	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF
SLOW_FWD	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SLOW_REV	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

processMediaDeviceRandomStateEvent			
State	**Channel 178**	**Channel 179**	**Channel 180**
RANDOM_DISC	ON	OFF	OFF
RANDOM_ALL	OFF	ON	OFF
RANDOM_OFF	OFF	OFF	ON

processMediaDeviceRepeatStateEvent				
State	**Channel 181**	**Channel 182**	**Channel 183**	**Channel 184**
REPEAT_DISC	ON	OFF	OFF	OFF
REPEAT_TRACK	OFF	ON	OFF	OFF
REPEAT_ALL	OFF	OFF	ON	OFF
REPEAT_OFF	OFF	OFF	OFF	ON

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Digital Satellite System

- IDigitalSatelliteSystemComponent
- IDigitalSatelliteSystemComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ITunerStationComponent
- ITunerStationComponentListener

Name: Digital Satellite System						
Interface: IDigitalSatelliteSystem						
Digital Satellite System Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleBand()	ITunerStationComponent	40			TUNER_BAND	Momentary Function Channel: Cycle tuner band when channel is activated
cycleDisplayInfo()	ITunerStationComponent	234			TUNER_OSD	Momentary Function Channel: Cycle on-screen or front-panel display info when channel is activated
cycleStationPresetGroup()	ITunerStationComponent	224			TUNER_PRESET_GROUP	Momentary Function Channel: Cycle station preset group/bank when channel is activated
decrementStation()	ITunerStationComponent	226			TUNER_STATION_DN	Momentary Function Channel: Station is decremented when channel is activated
getBand()	ITunerStationComponent			?BAND		Query for band, responds with BAND-<tb> where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getStation()	ITunerStationComponent			?XCH		Query for station, responds with XCH-<station> where <station> is a station string such as "501", "103.7" or "5.1"
getStationPreset()	ITunerStationComponent			?TUNERPRESET		Query for tuner preset, responds with TUNERPRESET-<preset>
getStationPresetCount()	ITunerStationComponent			?STATIONPRESET COUNT		Query for number of valid station presets, responds with STATIONPRESETCOUNT-<count>
getStationPresetProperties()	ITunerStationComponent			?STATIONPRESET PROPERTIES		Query input properties for all station presets, responds with STATIONPRESETPROPERTIES-<index>,<displayName>,<station> [;<index>,<displayName>,<station>]*
getStationPresetProperty(index)	ITunerStationComponent			?STATIONPRESET PROPERTY-<index>		Query properties for a single station preset, responds with STATIONPRESETPROPERTY-<index>, <displayName>,<station>
getStationPresetSelect()	ITunerStationComponent			?STATIONPRESET SELECT		Query for the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
getTunerBandProperties()	ITunerStationComponent			?TUNERBAND PROPERTIES		Query input properties for all tuner bands, responds with TUNERBANDPROPERTIES-<index>,<displayName>,<tb>[,<index>, <displayName>,<tb>]*
getTunerBandProperty(index)	ITunerStationComponent			?TUNERBAND PROPERTY-<index>		Query properties for a single tuner band, responds with TUNERBANDPROPERTY-<index>,<displayName>,<tb>
getTunerBandPropertyCount()	ITunerStationComponent			?TUNERBAND PROPERTYCOUNT		Query for number of valid tuner bands, responds with TUNERBANDPROPERTY-COUNT-<count>
getTunerBandSelect()	ITunerStationComponent			?TUNERBAND SELECT		Query for the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
getTunerComponentProperties()	ITunerStationComponent			?TUNERCOMPONENT PROPERTIES		Query properties for a single tuner component, responds with TUNERCOMPONENT-PROPERTIES-<index>,<displayName>, <value>[;<index>,<displayName>,<value>]*
getTunerComponentProperty()	ITunerStationComponent			?TUNERCOMPONENT PROPERTY		Query input properties for all inputs, responds with TUNERCOMPONENTPROPERTY-<index>,<displayName>,<value>
getTunerComponentPropertyCount()	ITunerStationComponent			?TUNERCOMPONENT PROPERTYCOUNT		Query for number of tuner components, responds with TUNERCOMPONENTPROPERTYCOUNT-<count>

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getTunerComponentSelect()	ITunerStationComponent			?TUNERCOMPONENT SELECT		Query for the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
gotoPreviousStation()	ITunerStationComponent	235			TUNER_PREV	Momentary Function Channel: Previous station is selected when channel is activated
incrementStation()	ITunerStationComponent	225			TUNER_STATION_UP	Momentary Function Channel: Station is incremented when channel is activated
nextStationPreset()	ITunerStationComponent	22			CHAN_UP	Momentary Function Channel: Next station preset is selected when channel is activated
previousStationPreset()	ITunerStationComponent	23			CHAN_DN	Momentary Function Channel: Previous station preset is selected when channel is activated
scanStation(FORWARD)	ITunerStationComponent	227			TUNER_SCAN_FWD	Momentary Function Channel: Scans for next station while channel is activate.
scanStation(BACKWARD)	ITunerStationComponent	228			TUNER_SCAN_REV	Momentary Function Channel: Scans for previous station while channel is activate
seekStation(FORWARD)	ITunerStationComponent	229			TUNER_SEEK_FWD	Momentary Function Channel: Seeks for next station while channel is activate
seekStation(BACKWARD)	ITunerStationComponent	230			TUNER_SEEK_REV	Momentary Function Channel: Seeks for previous station while channel is activate
setBand(tb)	ITunerStationComponent			BAND-<tb>		Set band, where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV
setStation(station)	ITunerStationComponent			XCH-<station>		Set station, where <station> is a station string such as "501", "103.7" or "5.1"
setStationPreset(preset)	ITunerStationComponent			TUNERPRESET-<preset>		Recall tuner preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setStationPresetSelect(index)	ITunerStationComponent			STATIONPRESET SELECT-<index>		Set the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
setTunerBandSelect(index)	ITunerStationComponent			TUNERBANDSELECT-<index>		Set the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
setTunerComponentSelect(index)	ITunerStationComponent			TUNERCOMPONENT SELECT-<index>		Set the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
processBandEvent(TunerBand)	ITunerStationComponentListener			BAND-<band>		Band changed, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO,LONG_WAVE,MEDIUM_WAVE,SHORT_WAVE, TV

Digital Satellite System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processStationEvent(String)	ITunerStationComponentListener			XCH-<station>		Station changed, where <station> is a station string such as "501", "103.7" or "5.1"
processStationPresetCountEvent(count)	ITunerStationComponentListener			STATIONPRESET COUNT-<count>		Response to ?TUNERCOUNT, where <count> is the integer number of tuner components
processStationPresetEvent(int)	ITunerStationComponentListener			TUNERPRESET-<int>		Tuner preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processStationPresetPropertiesEvent (String[])	ITunerStationComponentListener			STATIONPRESET PROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>, <value>;...]		Returns the properties for each station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetPropertyEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTY-<index>,<displayName>, <value>		Returns the properties for a single station pre-set, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetSelectEvent(index)	ITunerStationComponentListener			STATIONPRESET SELECT-<index>		Returns the selected station preset index
processTunerBandPropertiesEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTIES-<index>,<displayName>, <tunerBand>[;<index>, <displayName>, <tunerBand>;...]		Returns properties for each tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandPropertyCountEvent (count)	ITunerStationComponentListener			TUNERBANDPROPERTY COUNT-<count>		Returns the number of valid tuner bands
processTunerBandPropertyEvent(String[])	ITunerStationComponentListener			TUNERBANDPROPERTY-<index>,<displayName>, <tunerBand>		Returns properties for a single tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandSelectEvent(index)	ITunerStationComponentListener			TUNERBANDSELECT-<index>		Returns the selected tuner band index
processTunerComponentPropertiesEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTIES-<index>, <displayName>,<value> [;<index>,<displayName>, <value>;...]		Returns the properties for each tuner component
processTunerComponentPropertyCountEvent (count)	ITunerStationComponentListener			TUNERCOMPONENT PROPERTYCOUNT		Returns the integer number of tuner components
processTunerComponentPropertyEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single tuner component
processTunerComponentSelectEvent(index)	ITunerStationComponentListener			TUNERCOMPONENT SELECT-<index>		Returns the selected tuner component index

Digital Satellite System State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Digital Video Recorder

- IDigitalVideoRecorderComponent
- IDigitalVideoRecorderComponentListener
- IDiscTransportComponent
- IDiscTransportComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- ITunerStationComponent
- ITunerStationComponentListener

Name: Digital Video Recorder

Interface: IDigitalVideoRecorder

Digital Video Recorder Functions

Name	Component	Channel	Level	Command	Constant	Notes
cycleScanSpeed()	IDiscTransportComponent	192			SCAN_SPEED	Momentary Function Channel: Cycles the scan speed when the channel is activated
getTrackInfo()	IDiscTransportComponent			?TRACKINFO		Query for the track info, responds with TRACKINFO-<num>,<duration>, <discNumber>, where <duration> is a String in the format hh:mm:ss.ff
queryTrackProperties()	IDiscTransportComponent			?TRACKPROPS		Query for the track properties, responds with multiple TRACKPROP-<key>, <value> commands, one for each key
queryTrackProperty(key)	IDiscTransportComponent			?TRACKPROP-<keyName>		Query for a track property, responds with TRACKPROP-<key>,<value> command
setDiscTransport(PLAY)	IDiscTransportComponent	1			PLAY	Momentary Function Channel: Deck is set to play when the channel is activated
setDiscTransport(STOP)	IDiscTransportComponent	2			STOP	Momentary Function Channel: Deck is set to stop when the channel is activated
setDiscTransport(PAUSE)	IDiscTransportComponent	3			PAUSE	Momentary Function Channel: Deck is set to pause when the channel is activated
setDiscTransport(NEXT)	IDiscTransportComponent	4			FFWD	Momentary Function Channel: Deck is set to next track/chapter when the channel is activated

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setDiscTransport(PREVIOUS)	IDiscTransportComponent	5			REW	Momentary Function Channel: Deck is set to previous track/chapter when the channel is activated
setDiscTransport(SCAN_FWD)	IDiscTransportComponent	6			SFWD	Momentary Function Channel: Deck is set to scan forward when the channel is activated
setDiscTransport(SCAN_REV)	IDiscTransportComponent	7			SREV	Momentary Function Channel: Deck is set to scan reverse when the channel is activated
setDiscTransport(RECORD)	IDiscTransportComponent	8			RECORD	Momentary Function Channel: Deck is set to record when the channel is activated
setDiscTransport(FRAME_FWD)	IDiscTransportComponent	185			FRAME_FWD	Momentary Function Channel: Deck is set to step frame forward when the channel is activated
setDiscTransport(FRAME_REV)	IDiscTransportComponent	186			FRAME_REV	Momentary Function Channel: Deck is set to step frame reverse when the channel is activated
setDiscTransport(SLOW_FWD)	IDiscTransportComponent	188			SLOW_FWD	Momentary Function Channel: Deck is set to slow forward when the channel is activated
setDiscTransport(SLOW_REV)	IDiscTransportComponent	189			SLOW_REV	Momentary Function Channel: Deck is set to slow reverse when the channel is activated
setPlayPosition(mt)	IDiscTransportComponent			PLAYPOSITION-<counter>		Set the play position where <counter> is a String in the format [-]hh:mm:ss.ff, mm should be 0 >= mm < 60, ss should be 0 >= ss < 60, ff should be valid for the disc type
setTrackCounterNotificationOn(state)	IDiscTransportComponent			TRACKCOUNTER NOTIFY-<state>		Turn track counter notification on or off, where <state> is 1 or 0
processDiscTransportEvent(PLAY)	IDiscTransportComponentListener	241			PLAY_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(STOP)	IDiscTransportComponentListener	242			STOP_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(PAUSE)	IDiscTransportComponentListener	243			PAUSE_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(SCAN_FWD)	IDiscTransportComponentListener	246			SFWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(SCAN_REV)	IDiscTransportComponentListener	247			SREV_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(RECORD)	IDiscTransportComponentListener	248			RECORD_FB	Feedback Channel: Transport state change (see chart below)

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDiscTransportEvent(SLOW_FWD)	IDiscTransportComponentListener	249			SLOW_FWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent(SLOW_REV)	IDiscTransportComponentListener	250			SLOW_REV_FB	Feedback Channel: Transport state change (see chart below)
processTrackCounterEvent(MediaTime)	IDiscTransportComponentListener			TRACKCOUNTER-<counter>		Track counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processTrackInfoEvent(TrackInfo)	IDiscTransportComponentListener			TRACKINFO-<num>,<duration>, <discNumber>		Track info changed, where <num> is track number, <duration> is a track duration String in the format [-]hh:mm:ss.ff and <discNumber> is the disc number the track belongs to.
processTrackPropertiesEvent(Properties)	IDiscTransportComponentListener			TRACKPROP-<key>, <value>		Track properties query response, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs.

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName>[;<index>, <inputGroup>,<signalType>, <deviceLabel>,<displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputPropertiesEvent(String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName>[;<index>, <inputGroup>,<signalType>, <deviceLabel>, <displayName>]*		Returns comma-separated string containing <index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processInputPropertyEvent(String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cycleBand()	ITunerStationComponent	40			TUNER_BAND	Momentary Function Channel: Cycle tuner band when channel is activated
cycleDisplayInfo()	ITunerStationComponent	234			TUNER_OSD	Momentary Function Channel: Cycle on-screen or front-panel display info when channel is activated
cycleStationPresetGroup()	ITunerStationComponent	224			TUNER_PRESET_GROUP	Momentary Function Channel: Cycle station preset group/bank when channel is activated
decrementStation()	ITunerStationComponent	226			TUNER_STATION_DN	Momentary Function Channel: Station is decremented when channel is activated
getBand()	ITunerStationComponent			?BAND		Query for band, responds with BAND-<tb> where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV
getStation()	ITunerStationComponent			?XCH		Query for station, responds with XCH-<station> where <station> is a station string such as "501", "103.7" or "5.1"
getStationPreset()	ITunerStationComponent			?TUNERPRESET		Query for tuner preset, responds with TUNERPRESET-<preset>
getStationPresetCount()	ITunerStationComponent			?STATIONPRESETCOUNT		Query for number of valid station presets, responds with STATIONPRESETCOUNT-<count>
getStationPresetProperties()	ITunerStationComponent			?STATIONPRESET PROPERTIES		Query input properties for all station presets, responds with STATIONPRESETPROPERTIES-<index>,<displayName>,<station> [;<index>,<displayName>,<station>]*
getStationPresetProperty(index)	ITunerStationComponent			?STATIONPRESET PROPERTY-<index>		Query properties for a single station preset, responds with STATIONPRESETPROPERTY-<index>,<displayName>,<station>

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getStationPresetSelect()	ITunerStationComponent			?STATIONPRESETSELECT		Query for the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
getTunerBandProperties()	ITunerStationComponent			?TUNERBAND PROPERTIES		Query input properties for all tuner bands, responds with TUNERBANDPROPER-TIES-<index>,<displayName>, <tb>[,<index>,<displayName>,<tb>]*
getTunerBandProperty(index)	ITunerStationComponent			?TUNERBANDPROPERTY-<index>		Query properties for a single tuner band, responds with TUNERBANDPROPERTY-<index>,<displayName>,<tb>
getTunerBandPropertyCount()	ITunerStationComponent			?TUNERBANDPROPERTY COUNT		Query for number of valid tuner bands, responds with TUNERBANDPROPERTY-COUNT-<count>
getTunerBandSelect()	ITunerStationComponent			?TUNERBANDSELECT		Query for the currently selected index of valid tuner bands, responds with TUNER-BANDSELECT-<index>
getTunerComponentProperties()	ITunerStationComponent			?TUNERCOMPONENT PROPERTIES		Query properties for a single tuner compo-nent, responds with TUNERCOMPO-NENTPROPERTIES-<index>, <displayName>,<value>[;<index>, <displayName>,<value>]*
getTunerComponentProperty()	ITunerStationComponent			?TUNERCOMPONENT PROPERTY		Query input properties for all inputs, responds with TUNERCOMPONENT-PROPERTY-<index>,<displayName>, <value>
getTunerComponentPropertyCount()	ITunerStationComponent			?TUNERCOMPONENT PROPERTYCOUNT		Query for number of tuner components, responds with TUNERCOMPONENT-PROPERTYCOUNT-<count>
getTunerComponentSelect()	ITunerStationComponent			?TUNERCOMPONENT SELECT		Query for the currently selected index of tuner components, responds with TUNER-COMPONENTSELECT-<index>
gotoPreviousStation()	ITunerStationComponent	235			TUNER_PREV	Momentary Function Channel: Previous station is selected when channel is acti-vated
incrementStation()	ITunerStationComponent	225			TUNER_STATION_UP	Momentary Function Channel: Station is incremented when channel is activated
nextStationPreset()	ITunerStationComponent	22			CHAN_UP	Momentary Function Channel: Next sta-tion preset is selected when channel is activated
previousStationPreset()	ITunerStationComponent	23			CHAN_DN	Momentary Function Channel: Previous station preset is selected when channel is activated

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
scanStation(FORWARD)	ITunerStationComponent	227			TUNER_SCAN_FWD	Momentary Function Channel: Scans for next station while channel is activate.
scanStation(BACKWARD)	ITunerStationComponent	228			TUNER_SCAN_REV	Momentary Function Channel: Scans for previous station while channel is activate
seekStation(FORWARD)	ITunerStationComponent	229			TUNER_SEEK_FWD	Momentary Function Channel: Seeks for next station while channel is activate
seekStation(BACKWARD)	ITunerStationComponent	230			TUNER_SEEK_REV	Momentary Function Channel: Seeks for previous station while channel is activate
setBand(tb)	ITunerStationComponent			BAND-<tb>		Set band, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV
setStation(station)	ITunerStationComponent			XCH-<station>		Set station, where <station> is a station string such as "501", "103.7" or "5.1"
setStationPreset(preset)	ITunerStationComponent			TUNERPRESET-<preset>		Recall tuner preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setStationPresetSelect(index)	ITunerStationComponent			STATIONPRESETSELECT-<index>		Set the currently selected index of valid station presets, responds with STATION-PRESETSELECT-<index>
setTunerBandSelect(index)	ITunerStationComponent			TUNERBANDSELECT-<index>		Set the currently selected index of valid tuner bands, responds with TUNER-BANDSELECT-<index>
setTunerComponentSelect(index)	ITunerStationComponent			TUNERCOMPONENT SELECT-<index>		Set the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
processBandEvent(TunerBand)	ITunerStationComponentListener			BAND-<band>		Band changed, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV
processStationEvent(String)	ITunerStationComponentListener			XCH-<station>		Station changed, where <station> is a station string such as "501", "103.7" or "5.1"
processStationPresetCountEvent(count)	ITunerStationComponentListener			STATIONPRESETCOUNT-<count>		Response to ?TUNERCOUNT, where <count> is the integer number of tuner components
processStationPresetEvent(int)	ITunerStationComponentListener			TUNERPRESET-<int>		Tuner preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)

Digital Video Recorder Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processStationPresetPropertiesEvent (String[])	ITunerStationComponentListener			STATIONPRESET PROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>,<value>;...]		Returns the properties for each station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetPropertyEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetSelectEvent(index)	ITunerStationComponentListener			STATIONPRESETSELECT-<index>		Returns the selected station preset index
processTunerBandPropertiesEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTIES-<index>, <displayName>, <tunerBand>[;<index>, <displayName>, <tunerBand>;...]		Returns properties for each tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandPropertyCountEvent (count)	ITunerStationComponentListener			TUNERBAND PROPERTYCOUNT-<count>		Returns the number of valid tuner bands
processTunerBandPropertyEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTY-<index>, <displayName>, <tunerBand>		Returns properties for a single tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandSelectEvent(index)	ITunerStationComponentListener			TUNERBANDSELECT-<index>		Returns the selected tuner band index
processTunerComponentPropertiesEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTIES-<index>, <displayName>, <value>[;<index>, <displayName>,<value>;...]		Returns the properties for each tuner component
processTunerComponentPropertyCountEvent (count)	ITunerStationComponentListener			TUNERCOMPONENT PROPERTYCOUNT		Returns the integer number of tuner components
processTunerComponentPropertyEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single tuner component
processTunerComponentSelectEvent(index)	ITunerStationComponentListener			TUNERCOMPONENT SELECT-<index>		Returns the selected tuner component index

Digital Video Recorder State Charts

processDiscTransportEvent								
State	**Channel 241**	**Channel 242**	**Channel 243**	**Channel 246**	**Channel 247**	**Channel 248**	**Channel 249**	**Channel 250**
PLAY	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
STOP	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
SCAN_FWD	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SCAN_REV	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
RECORD	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
RECORD_PAUSE	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF
SLOW_FWD	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SLOW_REV	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Disc Device

- IDiscDeviceComponent
- IDiscDeviceComponentListener
- IDiscTransportComponent
- IDiscTransportComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener

Name: Disc Device

Interface: IDiscDevice

Disc Device Functions

Name	Component	Channel	Level	Command	Constant	Notes
cycleDiscTray()	IDiscDeviceComponent	120			DISC_TRAY	Momentary Function Channel: Open or Close the disc tray when the channel is activated
cycleRandom()	IDiscDeviceComponent	124			DISC_RANDOM	Momentary Function Channel: Cycle random when channel is activated
cycleRepeat()	IDiscDeviceComponent	125			DISC_REPEAT	Momentary Function Channel: Cycle repeat when channel is activated
getDiscCapacity()	IDiscDeviceComponent			?DISCCAPACITY		Query for the disc capacity responds with DISCCAPACITY-<discs> when <discs> is the number of disc slots supported by the device (see specific module documentation)
getDiscInfo()	IDiscDeviceComponent			?DISCINFO		Query for disc info, responds with DISCINFO-<num>,<duration>,<# of
getTitleInfo()	IDiscDeviceComponent			?TITLEINFO		Query for title info changed, responds with TITLEINFO-<num>,<duration>,<# of
nextDisc()	IDiscDeviceComponent	55			DISC_NEXT	Momentary Function Channel: Set discs next when channel is activated
previousDisc()	IDiscDeviceComponent	56			DISC_PREV	Momentary Function Channel: Set disc previous when channel is activated
queryDiscProperties()	IDiscDeviceComponent			?DISCPROPS		Query for the disc properties, responds with multiple DISCPROP-<key>,<value> commands, one for each key
queryDiscProperty(key)	IDiscDeviceComponent			?DISCPROP-<keyName>		Query for a disc property, responds with DISCPROP-<key>,<value> command
queryTitleProperties()	IDiscDeviceComponent			?TITLEPROPS		Query for the title properties, responds with multiple TITLEPROP-<key>,<value> commands, one for each key

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
queryTitleProperty(key)	IDiscDeviceComponent			?TITLEPROP-<keyName>		Query for a title property, responds with TITLEPROP-<key>,<value> command
setDisc(discNumber)	IDiscDeviceComponent			SETDISC-<discNumber>		Set disc number to <disc>, where <disc> is 1 to <x> where x is the disc capacity. (see getDiscCapactiy() or module specific documentation)
setPlayPosition (titleNumber,trackNumber)	IDiscDeviceComponent			PLAYPOSITION-<titleNumber>, <trackNumber>		Set the play position where <title> is the Title number and <track> is the Track number
setPlayPosition (titleNumber,trackNumber, relativeCounter)	IDiscDeviceComponent			PLAYPOSITION-<titleNumber>, <trackNumber>, <relativeCounter>		Set the play position where <title> is the Title number, <track> is the Track number and <counter> is a String in the format [-]hh:mm:ss.ff, mm should be 0 >= mm < 60, ss should be 0 >= ss < 60, ff should be valid for the disc type
setPlayPosition(trackNumber)	IDiscDeviceComponent			PLAYPOSITION-<trackNumber>		Set the play position where <track> is the Track number
setPlayPosition (trackNumber,relativeCounter)	IDiscDeviceComponent			PLAYPOSITION-<trackNumber>, <relativeCounter>		Set the play position where <track> is the Track number and <counter> is a String in the format [-]hh:mm:ss.ff, mm should be 0 >= mm < 60, ss should be 0 >= ss < 60, ff should be valid for the disc type
setRandomState (RANDOM_DISC)	IDiscDeviceComponent	178			DISC_RANDOM_DISC_ON	Momentary Function Channel: Random-disc is on while channel is active
setRandomState(RANDOM_ALL)	IDiscDeviceComponent	179			DISC_RANDOM_ALL_ON	Momentary Function Channel: Random-all is on while channel is active
setRandomState (RANDOM_OFF)	IDiscDeviceComponent	180			DISC_RANDOM_OFF_ON	Momentary Function Channel: Random-off is on while channel is active
setRepeatState(REPEAT_DISC)	IDiscDeviceComponent	181			DISC_REPEAT_DISC_ON	Momentary Function Channel: Repeat-disc is on while channel is active
setRepeatState (REPEAT_TRACK)	IDiscDeviceComponent	182			DISC_REPEAT_TRACK_ON	Momentary Function Channel: Repeat-track is on while channel is active
setRepeatState(REPEAT_ALL)	IDiscDeviceComponent	183			DISC_REPEAT_ALL_ON	Momentary Function Channel: Repeat-all is on while channel is active
setRepeatState(REPEAT_OFF)	IDiscDeviceComponent	184			DISC_REPEAT_OFF_ON	Momentary Function Channel: Repeat-off is on while channel is active
setTitleCounterNotificationOn (state)	IDiscDeviceComponent			TITLECOUNTER NOTIFY-<state>		Turn title counter notification on or off, where <state> is 1 or 0
processDiscInfoEvent(DiscInfo)	IDiscDeviceComponentListener			DISCINFO-<num>, <duration>,<totalTitles>, <totalTracks>, <discType>,<id>		Disc info changed, where <num> is disc number, <duration> is a disc duration String in the format [-]hh:mm:ss.ff, <# of tracks> is the number of titles in this disc, <# of tracks> is the number of tracks in this disc, <disctype> is the disc type (AUDIO_ONLY, VIDEO_ONLY, AUDIOVIDEO) and <id> is the disc database id.
processDiscPropertiesEvent (Properties)	IDiscDeviceComponentListener			DISCPROP-<key>, <value>		Disc properties query changed, responds with multiple DISCPROP-<key>,<value> commands, one for each key

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processRandomStateEvent (RANDOM_DISC)	IDiscDeviceComponentListener	178			DISC_RANDOM_DISC_FB	Feedback Channel: Random state change (see chart below)
processRandomStateEvent (RANDOM_ALL)	IDiscDeviceComponentListener	179			DISC_RANDOM_ALL_FB	Feedback Channel: Random state change (see chart below)
processRandomStateEvent (RANDOM_OFF)	IDiscDeviceComponentListener	180			DISC_RANDOM_OFF_FB	Feedback Channel: Random state change (see chart below)
processRepeatStateEvent (REPEAT_DISC)	IDiscDeviceComponentListener	181			DISC_REPEAT_DISC_FB	Feedback Channel: Repeat state change (see chart below)
processRepeatStateEvent (REPEAT_TRACK)	IDiscDeviceComponentListener	182			DISC_REPEAT_TRACK_FB	Feedback Channel: Repeat state change (see chart below)
processRepeatStateEvent (REPEAT_ALL)	IDiscDeviceComponentListener	183			DISC_REPEAT_ALL_FB	Feedback Channel: Repeat state change (see chart below)
processRepeatStateEvent (REPEAT_OFF)	IDiscDeviceComponentListener	184			DISC_REPEAT_OFF_FB	Feedback Channel: Repeat state change (see chart below)
processTitleCounterEvent (MediaTime)	IDiscDeviceComponentListener			TITLECOUNTER-<counter>		Title counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processTitleInfoEvent(TitleInfo)	IDiscDeviceComponentListener			TITLEINFO-<num>, <duration>, <# of tracks>, <discNumber>		Title info changed, where <num> is title number, <duration> is a title duration String in the format [-]hh:mm:ss.ff, <# of tracks> is the number of tracks in this title and <discNumber> is the disc number the title belongs to.
processTitlePropertiesEvent (Properties)	IDiscDeviceComponentListener			TITLEPROP-<key>, <value>		Title properties query response, responds with multiple TITLEPROP-<key>,<value> commands, one for each key
cycleScanSpeed()	IDiscTransportComponent	192			SCAN_SPEED	Momentary Function Channel: Cycles the scan speed when the channel is activated
getTrackInfo()	IDiscTransportComponent			?TRACKINFO		Query for the track info, responds with TRACKINFO-<num>,<duration>,<discNumber>, where <duration> is a String in the format hh:mm:ss.ff
queryTrackProperties()	IDiscTransportComponent			?TRACKPROPS		Query for the track properties, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
queryTrackProperty(key)	IDiscTransportComponent			?TRACKPROP-<keyName>		Query for a track property, responds with TRACKPROP-<key>,<value> command
setDiscTransport(PLAY)	IDiscTransportComponent	1			PLAY	Momentary Function Channel: Deck is set to play when the channel is activated
setDiscTransport(STOP)	IDiscTransportComponent	2			STOP	Momentary Function Channel: Deck is set to stop when the channel is activated
setDiscTransport(PAUSE)	IDiscTransportComponent	3			PAUSE	Momentary Function Channel: Deck is set to pause when the channel is activated

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setDiscTransport(NEXT)	IDiscTransportComponent	4			FFWD	Momentary Function Channel: Deck is set to next track/ chapter when the channel is activated
setDiscTransport(PREVIOUS)	IDiscTransportComponent	5			REW	Momentary Function Channel: Deck is set to previous track/chapter when the channel is activated
setDiscTransport(SCAN_FWD)	IDiscTransportComponent	6			SFWD	Momentary Function Channel: Deck is set to scan forward when the channel is activated
setDiscTransport(SCAN_REV)	IDiscTransportComponent	7			SREV	Momentary Function Channel: Deck is set to scan reverse when the channel is activated
setDiscTransport(RECORD)	IDiscTransportComponent	8			RECORD	Momentary Function Channel: Deck is set to record when the channel is activated
setDiscTransport(FRAME_FWD)	IDiscTransportComponent	185			FRAME_FWD	Momentary Function Channel: Deck is set to step frame forward when the channel is activated
setDiscTransport(FRAME_REV)	IDiscTransportComponent	186			FRAME_REV	Momentary Function Channel: Deck is set to step frame reverse when the channel is activated
setDiscTransport(SLOW_FWD)	IDiscTransportComponent	188			SLOW_FWD	Momentary Function Channel: Deck is set to slow forward when the channel is activated
setDiscTransport(SLOW_REV)	IDiscTransportComponent	189			SLOW_REV	Momentary Function Channel: Deck is set to slow reverse when the channel is activated
setPlayPosition(mt)	IDiscTransportComponent			PLAYPOSITION-<counter>		Set the play position where <counter> is a String in the format [-]hh:mm:ss.ff, mm should be 0 >= mm < 60, ss should be 0 >= ss < 60, ff should be valid for the disc type
setTrackCounterNotificationOn (state)	IDiscTransportComponent			TRACKCOUNTER NOTIFY-<state>		Turn track counter notification on or off, where <state> is 1 or 0
processDiscTransportEvent (PLAY)	IDiscTransportComponentListener	241			PLAY_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (STOP)	IDiscTransportComponentListener	242			STOP_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (PAUSE)	IDiscTransportComponentListener	243			PAUSE_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SCAN_FWD)	IDiscTransportComponentListener	246			SFWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SCAN_REV)	IDiscTransportComponentListener	247			SREV_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (RECORD)	IDiscTransportComponentListener	248			RECORD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SLOW_FWD)	IDiscTransportComponentListener	249			SLOW_FWD_FB	Feedback Channel: Transport state change (see chart below)
processDiscTransportEvent (SLOW_REV)	IDiscTransportComponentListener	250			SLOW_REV_FB	Feedback Channel: Transport state change (see chart below)

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processTrackCounterEvent (MediaTime)	IDiscTransportComponentListener			TRACKCOUNTER-<counter>		Track counter changed, where <counter> is a String in the format [-]hh:mm:ss.ff
processTrackInfoEvent(TrackInfo)	IDiscTransportComponentListener			TRACKINFO-<num>, <duration>, <discNumber>		Track info changed, where <num> is track number, <duration> is a track duration String in the format [-]hh:mm:ss.ff and <discNumber> is the disc number the track belongs to.
processTrackPropertiesEvent (Properties)	IDiscTransportComponentListener			TRACKPROP-<key>, <value>		Track properties query response, responds with multiple TRACKPROP-<key>,<value> commands, one for each key
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor (DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton (THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton (THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FOR-WARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton (RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton (DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton (ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTER-ISK
pressMenuButton (DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton (POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton (COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton (DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton (CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton (PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton (PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton (DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton (SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton (INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton (FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton (CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton (RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton (GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton (PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton (DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton (TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton (PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton (AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton (HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton (TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton (TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton (ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton (ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton (AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton (PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton (SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton (SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton (FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton (RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton (INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent (byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated

Disc Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>, <channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Disc Device State Charts

processRandomStateEvent			
State	**Channel 178**	**Channel 179**	**Channel 180**
RANDOM_DISC	ON	OFF	OFF
RANDOM_ALL	OFF	ON	OFF
RANDOM_OFF	OFF	OFF	ON

processRepeatStateEvent				
State	**Channel 181**	**Channel 182**	**Channel 183**	**Channel 184**
REPEAT_DISC	ON	OFF	OFF	OFF
REPEAT_TRACK	OFF	ON	OFF	OFF
REPEAT_ALL	OFF	OFF	ON	OFF
REPEAT_OFF	OFF	OFF	OFF	ON

processDiscTransportEvent								
State	**Channel 241**	**Channel 242**	**Channel 243**	**Channel 246**	**Channel 247**	**Channel 248**	**Channel 249**	**Channel 250**
PLAY	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
STOP	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
SCAN_FWD	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SCAN_REV	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
RECORD	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
RECORD_PAUSE	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF
SLOW_FWD	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SLOW_REV	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Document Camera

- IDocumentCameraComponent
- IDocumentCameraComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener

Name: Document Camera						
Interface: IDocumentCamera						
Document Camera Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleAutoFocus()	IDocumentCameraComponent	172			AUTO_FOCUS	Momentary Function Channel: Cycle auto focus when channel is activated
cycleAutoIris()	IDocumentCameraComponent	173			AUTO_IRIS	Momentary Function Channel: Cycle auto iris when channel is activated
cycleLight()	IDocumentCameraComponent	176			DOCCAM_LIGHT	Momentary Function Channel: Cycle lights when channel is activated
setAutoFocusOn(state)	IDocumentCameraComponent	162			AUTO_FOCUS_ON	Discrete Function Channel: Auto focus is on while channel is active
setAutoIrisOn(state)	IDocumentCameraComponent	163			AUTO_IRIS_ON	Discrete Function Channel: Auto iris is on while channel is active
setFocus(focus)	IDocumentCameraComponent		16		FOCUS_LVL	Set focus position, range is 0-255 (0=near)
setFocusRamp(NEAR)	IDocumentCameraComponent	160			FOCUS_NEAR	Ramping Channel: Focus is ramped near while channel is active
setFocusRamp(FAR)	IDocumentCameraComponent	161			FOCUS_FAR	Ramping Channel: Focus is ramped far while channel is active
setFocusSpeed(speed)	IDocumentCameraComponent		19		FOCUS_SPEED_LVL	Set focus speed, range is 0-255 (0=slow)
setIris(iris)	IDocumentCameraComponent		17		IRIS_LVL	Set iris position, range is 0-255 (0=closed)
setIrisRamp(OPEN)	IDocumentCameraComponent	174			IRIS_OPEN	Ramping Channel: Iris is ramped open while channel is active
setIrisRamp(CLOSE)	IDocumentCameraComponent	175			IRIS_CLOSE	Ramping Channel: Iris is ramped closed while channel is active
setIrisSpeed(speed)	IDocumentCameraComponent		20		IRIS_SPEED_LVL	Set iris speed, range is 0-255 (0=slow)
setLowerLightOn(state)	IDocumentCameraComponent	197			DOCCAM_LOWER_LIGHT_ON	Discrete Function Channel: Lower light is on while channel is active
setUpperLightOn(state)	IDocumentCameraComponent	198			DOCCAM_UPPER_LIGHT_ON	Discrete Function Channel: Upper light is on while channel is active

Document Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setZoom(zoom)	IDocumentCameraComponent		15		ZOOM_LVL	Set zoom position, range is 0-255 (0=out/wide)
setZoomRamp(OUT)	IDocumentCameraComponent	158			ZOOM_OUT	Ramping Channel: Zoom is ramped out (far) while channel is active
setZoomRamp(IN)	IDocumentCameraComponent	159			ZOOM_IN	Ramping Channel: Zoom is ramped in (tele) while channel is active
setZoomSpeed(speed)	IDocumentCameraComponent		18		ZOOM_SPEED_LVL	Set zoom speed, range is 0-255 (0=slow)
processAutoFocusEvent (boolean)	IDocumentCameraComponentListener	162			AUTO_FOCUS_FB	Feedback Channel: Auto focus is on if channel is active
processAutoIrisEvent (boolean)	IDocumentCameraComponentListener	163			AUTO_IRIS_FB	Feedback Channel: Auto iris is on if channel is active
processFocusEvent(int)	IDocumentCameraComponentListener		16		FOCUS_LVL	Focus changed, range is 0-255 (0=near)
processFocusRampEvent (NEAR)	IDocumentCameraComponentListener	160			FOCUS_NEAR_FB	Feedback Channel: Focus is ramping near while channel is on
processFocusRampEvent (FAR)	IDocumentCameraComponentListener	161			FOCUS_FAR_FB	Feedback Channel: Focus is ramping far while channel is on
processFocusSpeedEvent(int)	IDocumentCameraComponentListener		19		FOCUS_SPEED_LVL	Focus speed changed, range is 0-255 (0=slow)
processIrisEvent(int)	IDocumentCameraComponentListener		17		IRIS_LVL	Iris changed, range is 0-255 (0=closed)
processIrisRampEvent (OPEN)	IDocumentCameraComponentListener	174			IRIS_OPEN_FB	Feedback Channel: Iris is ramping open while channel is on
processIrisRampEvent (CLOSE)	IDocumentCameraComponentListener	175			IRIS_CLOSE_FB	Feedback Channel: Iris is ramping closed while channel is on
processIrisSpeedEvent(int)	IDocumentCameraComponentListener		20		IRIS_SPEED_LVL	Iris speed changed, range is 0-255 (0=slow)
processLowerLightEvent (boolean)	IDocumentCameraComponentListener	197			DOCCAM_LOWER_LIGHT_FB	Feedback Channel: Lower light is on if channel is active
processUpperLightEvent (boolean)	IDocumentCameraComponentListener	198			DOCCAM_UPPER_LIGHT_FB	Feedback Channel: Upper light is on if channel is active
processZoomEvent(int)	IDocumentCameraComponentListener		15		ZOOM_LVL	Zoom changed, range is 0-255 (0=out/wide)
processZoomRampEvent (OUT)	IDocumentCameraComponentListener	158			ZOOM_OUT_FB	Feedback Channel: Zoom is ramping out (wide) while channel is on
processZoomRampEvent(IN)	IDocumentCameraComponentListener	159			ZOOM_IN_FB	Feedback Channel: Zoom is ramping in (tele) while channel is on
processZoomSpeedEvent(int)	IDocumentCameraComponentListener		18		ZOOM_SPEED_LVL	Zoom speed changed, range is 0-255 (0=slow)

Document Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent (byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated

Document Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent (count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs

Document Camera Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processInputPropertiesEvent (String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>, <inputGroup>,<signalType>, <deviceLabel>,<displayName> [;<index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName>]*		Returns comma-separated string containing <index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputPropertyEvent (String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>, <inputGroup>,<signalType>, <deviceLabel>,<displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent (index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input

Document Camera State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

HVAC

- IHVACComponent
- IHVACComponentListener
- IModuleComponent
- IModuleComponentListener

Name: HVAC						
Interface: IHVAC						
HVAC Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
addHVACComponent (index,hvacAddress)	IHVACComponent			HVACADD-<index>, <hvacAddress>		Add a thermostat at a given index, where <index> is 1-x and <address> is a thermostat address and x is the maximum supported thermostat index (see specific module documentation)
cycleFanState()	IHVACComponent	213			HVAC_FAN	Momentary Function Channel: Cycle the fan state when channel is activated
cycleHumidifyState()	IHVACComponent	217			HVAC_HUMIDIFY_STATE	Momentary Function Channel: Cycle the humidify state when channel is activated
cycleHVACState()	IHVACComponent	218			HVAC_STATE	Momentary Function Channel: Cycle the HVAC state when channel is activated
decrementCoolSetpoint()	IHVACComponent	141			HVAC_COOL_DN	Momentary Function Channel: Decrement the cool setpoint when channel is activated
decrementDehumidifySetpoint()	IHVACComponent	151			HVAC_DEHUMIDIFY_DN	Momentary Function Channel: Decrement the dehumidify setpoint when channel is activated
decrementHeatSetpoint()	IHVACComponent	144			HVAC_HEAT_DN	Momentary Function Channel: Decrement the heat setpoint when channel is activated
decrementHumidifySetpoint()	IHVACComponent	149			HVAC_HUMIDIFY_DN	Momentary Function Channel: Decrement the humidify setpoint when channel is activated
getHumidifyState()	IHVACComponent			?HVACHUMID		Query for the humidify state, responds with HVACHUMID-<state> where <state> is OFF,HUMIDIFY, DEHUMIDIFY,AUTO
getHumidifyStatus()	IHVACComponent			?HVACHUMIDSTATUS		Query for the humidify status, responds with HVACHUMIDSTATUS-<status> where <status> is OFF,HUMIDIFY,DEHUMIDIFY
getHVACComponentAddress (index)	IHVACComponent			?HVACADDR-<index>		Query for the address of the thermostat at index <index>, responds with HVACADDR-<index>,<hvacAddress>
getHVACComponentIndex (hvacAddress)	IHVACComponent			?HVACIDX-<hvacAddress>		Query for the index of the thermostat with address <hvacAddress>, responds with HVACADDR-<index>, <hvacAddress>
getTemperatureScale()	IHVACComponent			?HVACSCALE		Query for the temperature scale, responds with HVACSCALE-<scale> where <scale> is FAHRENHEIT, CELSIUS

HVAC Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
incrementCoolSetpoint()	IHVACComponent	140			HVAC_COOL_UP	Momentary Function Channel: Increment the cool setpoint when channel is activated
incrementDehumidifySetpoint()	IHVACComponent	150			HVAC_DEHUMIDIFY_UP	Momentary Function Channel: Increment the dehumidify setpoint when channel is activated
incrementHeatSetpoint()	IHVACComponent	143			HVAC_HEAT_UP	Momentary Function Channel: Increment the heat setpoint when channel is activated
incrementHumidifySetpoint()	IHVACComponent	148			HVAC_HUMIDIFY_UP	Momentary Function Channel: Increment the humidify setpoint when channel is activated
removeHVACComponent (hvacAddress)	IHVACComponent			HVACREMOVEADDR-<hvacAddress>		Remove the thermostat with address <hvacAddress>, where <hvacAddress> is a thermostat address
removeHVACComponent(index)	IHVACComponent			HVACREMOVEIDX-<index>		Remove the thermostat at index <index>, where <index> is 1-x and x is the maximum supported thermostat index (see specific module documentation)
setCoolSetpoint(setpoint)	IHVACComponent		31		HVAC_COOL_LVL	Set the cool setpoint, value is in degrees C or F depending on temperature scale
setDehumidifySetpoint(setpoint)	IHVACComponent		38		HVAC_DEHUMIDIFY_LVL	Set the dehumidify setpoint, value is in percent
setFanState(ON)	IHVACComponent	214			HVAC_FAN_ON	Momentary Function Channel: Fan state is on while channel is active
setFanState(AUTO)	IHVACComponent	215			HVAC_FAN_AUTO	Momentary Function Channel: Fan state is auto while channel is active
setHeatSetpoint(setpoint)	IHVACComponent		32		HVAC_HEAT_LVL	Set the heat setpoint, value is in degrees C or F depending on temperature scale
setHoldOn(state)	IHVACComponent	211			HVAC_HOLD_ON	Discrete Function Channel: Thermostat hold mode is on while channel is active
setHumidifySetpoint(setpoint)	IHVACComponent		37		HVAC_HUMIDIFY_LVL	Set the humidify setpoint, value is in percent
setHumidifyState(AUTO)	IHVACComponent	228			HVAC_HUMIDIFY_AUTO	Momentary Function Channel: Humidity state is auto while channel is active
setHumidifyState(DEHUMIDIFY)	IHVACComponent	229			HVAC_DEHUMIDIFY	Momentary Function Channel: Humidity state is dehumidify while channel is active
setHumidifyState(HUMIDIFY)	IHVACComponent	230			HVAC_HUMIDIFY	Momentary Function Channel: Humidity state is humidify while channel is active
setHumidifyState(OFF)	IHVACComponent	231			HVAC_HUMIDIFY_OFF	Momentary Function Channel: Humidity state is off while channel is active
setHumidifyState(hs)	IHVACComponent			HVACHUMID-<hs>		Set the humidify state, where <hs> is OFF,HUMIDIFY, DEHUMIDIFY,AUTO
setHVACState(AUTO)	IHVACComponent	219			HVAC_AUTO	Momentary Function Channel: HVAC state is auto while channel is active

HVAC Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setHVACState(COOL)	IHVACComponent	220			HVAC_COOL	Momentary Function Channel: HVAC state is cool while channel is active
setHVACState(HEAT)	IHVACComponent	221			HVAC_HEAT	Momentary Function Channel: HVAC state is heat while channel is active
setHVACState(OFF)	IHVACComponent	222			HVAC_OFF	Momentary Function Channel: HVAC state is off while channel is active
setHVACState (EMERGENCY_HEAT)	IHVACComponent	223			HVAC_EHEAT	Momentary Function Channel: HVAC state is emergency heat while channel is active
setLockOn(state)	IHVACComponent	212			HVAC_LOCK_ON	Discrete Function Channel: Thermostat is locked while channel is active
setTemperatureScale(ts)	IHVACComponent			HVACSCALE-<ts>		Set the temperature scale, where <ts> is FAHRENHEIT, CELSIUS
processCoolSetpointEvent(int)	IHVACComponentListener		31		HVAC_COOL_LVL	Cool setpoint changed, value is in degrees C or F depending on temperature scale
processDehumidifySetpointEvent (int)	IHVACComponentListener		38		HVAC_DEHUMIDIFY_LVL	Dehumidify setpoint changed, value is in percent
processFanStateEvent(ON)	IHVACComponentListener	214			HVAC_FAN_ON_FB	Feedback Channel: Fan state is on while channel is on
processFanStateEvent(AUTO)	IHVACComponentListener	215			HVAC_FAN_AUTO_FB	Feedback Channel: Fan state is Auto while channel is on
processHeatSetpointEvent(int)	IHVACComponentListener		32		HVAC_HEAT_LVL	Heat setpoint changed, value is in degrees C or F depending on temperature scale
processHoldEvent(boolean)	IHVACComponentListener	211			HVAC_HOLD_FB	Feedback Channel: Thermostat hold mode is on while channel is on
processHumidifySetpointEvent (int)	IHVACComponentListener		37		HVAC_HUMIDIFY_LVL	Humidify setpoint changed, value is in percent
processHumidifyStateEvent (AUTO)	IHVACComponentListener	228			HVAC_HUMIDIFY_AUTO_FB	Feedback Channel: Humidity state change (see chart below)
processHumidifyStateEvent (DEHUMIDIFY)	IHVACComponentListener	229			HVAC_DEHUMIDIFY_FB	Feedback Channel: Humidity state change (see chart below)
processHumidifyStateEvent (HUMIDIFY)	IHVACComponentListener	230			HVAC_HUMIDIFY_FB	Feedback Channel: Humidity state change (see chart below)
processHumidifyStateEvent (OFF)	IHVACComponentListener	231			HVAC_HUMIDIFY_OFF_FB	Feedback Channel: Humidity state change (see chart below)
processHumidifyStateEvent (HumidifyState)	IHVACComponentListener			HVACHUMID-<state>		Humidify state changed, <state> is OFF,HUMIDIFY, DEHUMIDIFY,AUTO
processHumidifyStatusEvent (DEHUMIDIFY)	IHVACComponentListener	232			HVAC_DEHUMIDIFING_FB	Feedback Channel: Humidity status change (see chart below)
processHumidifyStatusEvent (HUMIDIFY)	IHVACComponentListener	233			HVAC_HUMIDIFING_FB	Feedback Channel: Humidity status change (see chart below)

HVAC Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processHumidifyStatusEvent (HumidifyStatus)	IHVACComponentListener			HVACHUMIDSTATUS-<status>		Humidify status changed, <status> is OFF,HUMIDIFY, DEHUMIDIFY
processHVACStateEvent (AUTO)	IHVACComponentListener	219			HVAC_AUTO_FB	Feedback Channel: HVAC state change (see chart below)
processHVACStateEvent(COOL)	IHVACComponentListener	220			HVAC_COOL_FB	Feedback Channel: HVAC state change (see chart below)
processHVACStateEvent(HEAT)	IHVACComponentListener	221			HVAC_HEAT_FB	Feedback Channel: HVAC state change (see chart below)
processHVACStateEvent(OFF)	IHVACComponentListener	222			HVAC_OFF_FB	Feedback Channel: HVAC state change (see chart below)
processHVACStateEvent (EMERGENCY_HEAT)	IHVACComponentListener	223			HVAC_EHEAT_FB	Feedback Channel: HVAC state change (see chart below)
processHVACStatusEvent (COOL)	IHVACComponentListener	224			HVAC_COOLING_FB	Feedback Channel: HVAC status change (see chart below)
processHVACStatusEvent (HEAT)	IHVACComponentListener	225			HVAC_HEATING_FB	Feedback Channel: HVAC status change (see chart below)
processHVACStatusEvent (COOL_2)	IHVACComponentListener	226			HVAC_COOLING2_FB	Feedback Channel: HVAC status change (see chart below)
processHVACStatusEvent (EMERGENCY_HEAT)	IHVACComponentListener	227			HVAC_EHEATING_FB	Feedback Channel: HVAC status change (see chart below)
processIndoorHumidityEvent(int)	IHVACComponentListener		35		INDOOR_HUMID_LVL	Indoor humidity changed, value is in percent
processIndoorTemperatureEvent (float)	IHVACComponentListener		33		INDOOR_TEMP_LVL	Indoor temperature changed, value is in degrees C or F depending on temperature scale
processLockEvent(boolean)	IHVACComponentListener	212			HVAC_LOCK_FB	Feedback Channel: Thermostat is locked while channel is on
processOutdoorHumidityEvent (int)	IHVACComponentListener		36		OUTDOOR_HUMID_LVL	Outdoor humidity changed, value is in percent
processOutdoorTemperatureEvent (float)	IHVACComponentListener		34		OUTDOOR_TEMP_LVL	Outdoor temperature changed, value is in degrees C or F depending on temperature scale
processTemperatureScaleEvent (TemperatureScale)	IHVACComponentListener			HVACSCALE-<scale>		HVAC scale changed, <scale> is FAHRENHEIT,CELSIUS
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.

HVAC Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

HVAC Listener State Charts

processFanStateEvent		
State	**Channel 214**	**Channel 215**
ON	ON	OFF
AUTO	OFF	ON

processFanStatusEvent	
State	**Channel 216**
OFF	OFF
ON	ON

processHumidifyStateEvent				
State	**Channel 228**	**Channel 229**	**Channel 230**	**Channel 231**
AUTO	ON	OFF	OFF	OFF
DEHUMIDIFY	OFF	ON	OFF	OFF
HUMIDIFY	OFF	OFF	ON	OFF
OFF	OFF	OFF	OFF	ON

processHumidifyStatusEvent		
State	**Channel 232**	**Channel 233**
OFF	OFF	OFF
DEHUMIDIFY	ON	OFF
HUMIDIFY	OFF	ON

processHVACStateEvent					
State	**Channel 219**	**Channel 220**	**Channel 221**	**Channel 222**	**Channel 223**
AUTO	ON	OFF	OFF	OFF	OFF
COOL	OFF	ON	OFF	OFF	OFF
HEAT	OFF	OFF	ON	OFF	OFF
OFF	OFF	OFF	OFF	ON	OFF
EMERGENCY_HEAT	OFF	OFF	OFF	OFF	ON

processHVACStatusEvent				
State	**Channel 224**	**Channel 225**	**Channel 226**	**Channel 227**
OFF	OFF	OFF	OFF	OFF
COOL	ON	OFF	OFF	OFF
HEAT	OFF	ON	OFF	OFF
COOL_2	OFF	OFF	ON	OFF
EMERGENCY_HEAT	OFF	OFF	OFF	ON

IO Device

- IIODeviceComponent
- IIODeviceComponentListener
- IModuleComponent
- IModuleComponentListener

Name: IO Device						
Interface: IIODevice						
IO Device Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getIOChannelCount()	IIODeviceComponent			?IOCHANNELCOUNT		Query for the number of channels on an IO device, responds with IOCHANNELCOUNT
getIOChannelDirection(io-chan)	IIODeviceComponent			?IOCHANNELDIRECTION-<channel>		Query for direction of the I/O channel, where <io-chan> is the integer channel number, responds with IOCHANNELDIRECTION
getIOChannelInputSense(io-chan)	IIODeviceComponent			?IOCHANNELINPUTSENSE-<channel>		Query for input sense of the I/O channel, where <io-chan> is the integer channel number, responds with IOCHANNELINPUTSENSE
getIOChannelState(io-chan)	IIODeviceComponent			?IOCHANNELSTATE-<channel>		Query for state of the I/O channel, where <io-chan> is the integer channel number, responds with IOCHANNELSTATE
setIOChannelDirection (io-chan,io-dir)	IIODeviceComponent			IOCHANNELDIRECTION-<channel>,<direction>		Sets the I/O channel direction, where <io-chan> is the integer channel number and <io-dir> is INPUT or OUTPUT
setIOChannelInputSense (io-chan,io-sense)	IIODeviceComponent			IOCHANNELINPUTSENSE-<channel>,<sense>		Sets the I/O channel input sense, where <io-chan> is the integer channel number and <io-sense> is HIGH or LOW
setIOChannelState(io-chan,io-state)	IIODeviceComponent			IOCHANNELSTATE-<channel>,<state>		Sets the I/O channel state, where <io-chan> is the integer channel number and <io-state> is ON or OFF
processIOChannelCountEvent (count)	IIODeviceComponentListener			IOCHANNELCOUNT-<count>		Response to ?IOCHANNELCOUNT, where <count> is an integer value
processIOChannelDirectionEvent (io-dir)	IIODeviceComponentListener			IOCHANNELDIRECTION-<channel>,<direction>		Response to ?IOCHANNELDIRECTION, where <io-chan> is the integer channel number and <io-dir> is INPUT or OUTPUT
processIOChannelInputSenseEvent (io-sense)	IIODeviceComponentListener			IOCHANNELINPUTSENSE-<channel>,<sense>		Response to ?IOCHANNELINPUTSENSE, where <io-chan> is the integer channel number and <io-sense> is HIGH or LOW
processIOChannelStateEvent (io-state)	IIODeviceComponentListener			IOCHANNELSTATE-<channel>,<state>		Response to ?IOCHANNELSTATE, where <io-chan> is the integer channel number and <io-state> is ON or OFF
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO

IO Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy><hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

Keypad

- IKeypadComponent
- IKeypadComponentListener
- IModuleComponent
- IModuleComponentListener

Name: Keypad						
Interface: IKeypad						
Keypad Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
addKeypadComponent (index,keypadAddress)	IKeypadComponent			KEYPADADD-<index>, <keypadAddress>		Add a keypad address at a given index, where <index> is 1-x and <keypadaddr> is a keypad address and x is the maximum supported keypad index (see specific module documentation)
getKeypadComponentAddress (index)	IKeypadComponent			?KEYPADADDR-<index>		Query for the address of the keypad at index <index>, responds with KEYPADADDR-<index>,<keypadaddr>
getKeypadComponentIndex (keypadAddress)	IKeypadComponent			?KEYPADIDX-<keypadaddr>		Query for the index of the keypad with address <keypadaddr>, responds with KEYPADADDR-<index>,<keypadaddr>
removeKeypadComponent (keypadAddress)	IKeypadComponent			KEYPADREMOVEADDR-<keypadAddress>		Remove the keypad with address <keypadaddr>, where <keypadaddr> is a keypad address
removeKeypadComponent (index)	IKeypadComponent			KEYPADREMOVEIDX-<index>		Remove the keypad at index <index>, where <index> is 1-x and x is the maximum supported keypad index (see specific module documentation)
setButtonState(btnNum,bs)	IKeypadComponent			KEYPADBTN-<btnAddr>,<state>		Set the state of a keypad button <btn> for the keypad at index/port, where <state> is CLICK or DOUBLECLICK
setButtonStatus(btnNum,bs)	IKeypadComponent	<btn>				Discrete Function Channel: Set the status of a keypad button <btn> for the keypad at index/port (see state chart)
setButtonStatus(btnNum,bs)	IKeypadComponent	<btn>+100				Discrete Function Channel: Set the status of a keypad button <btn> for the keypad at index/port (see state chart)
processButtonStateEvent (PUSH)	IKeypadComponentListener	<btn>				Input Function Channel: Button state is Released when BUTTON_EVENT-RELEASE is received, where <btn> is the button/channel number
processButtonStateEvent (RELEASE)	IKeypadComponentListener	<btn>				Input Function Channel: Button state is Pushed when BUTTON_EVENT-PUSH is received, where <btn> is the button/channel number
processButtonStateEvent (ButtonState)	IKeypadComponentListener			KEYPADBTN-<btn>,<state>		Button State changed. <btn> is button number and <state> is CLICK or DOUBLECLICK
processButtonStatusEvent (ON)	IKeypadComponentListener	<btn>				Feedback Channel: Button status on ON while channel <btn> is on and channel <btn>+100 is off, where <btn> is the button/channel number (see chart below)
processButtonStatusEvent (BLINK)	IKeypadComponentListener	<btn>+100				Feedback Channel: Button status on BLINK while channel <btn> is on and channel <btn>+100 is on, where <btn> is the button/channel number (see chart below)

Keypad Functions (Con.t)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

Keypad Listener State Charts

processButtonStateEvent	
State	**Channel <btn>**
RELEASE	OFF
PUSH	ON

processButtonStatusEvent		
State	**Channel <btn>**	**Channel <btn>+100**
OFF	OFF	OFF
ON	ON	OFF
BLINK	ON	ON
Note: *All Keypad commands include an index. This index is used to obtain the keypad component and the function is called on that component.* *The range for channel '<btn>' is 1 to 100; the corresponding range for channel '<btn> + 100' is 101 to 200.* *Channels above 200 are passed to the module for advanced processing.*		

Light

- IKeypadComponent
- IKeypadComponentListener
- ILightComponent
- ILightComponentListener
- IModuleComponent
- IModuleComponentListener

Name: Light						
Interface: ILight						
Light Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
addKeypadComponent (index,keypadAddress)	IKeypadComponent			KEYPADADD-<index>, <keypadAddress>		Add a keypad address at a given index, where <index> is 1-x and <keypadaddr> is a keypad address and x is the maximum supported keypad index (see specific module documentation)
getKeypadComponentAddress (index)	IKeypadComponent			?KEYPADADDR-<index>		Query for the address of the keypad at index <index>, responds with KEYPADADDR-<index>,<keypadaddr>
getKeypadComponentIndex (keypadAddress)	IKeypadComponent			?KEYPADIDX-<keypadaddr>		Query for the index of the keypad with address <keypadaddr>, responds with KEYPADADDR-<index>,<keypadaddr>
removeKeypadComponent (keypadAddress)	IKeypadComponent			KEYPADREMOVEADDR-<keypadAddress>		Remove the keypad with address <keypadaddr>, where <keypadaddr> is a keypad address
removeKeypadComponent (index)	IKeypadComponent			KEYPADREMOVEIDX-<index>		Remove the keypad at index <index>, where <index> is 1-x and x is the maximum supported keypad index (see specific module documentation)
setButtonState(btnNum,bs)	IKeypadComponent			KEYPADBTN-<btnAddr>, <state>		Set the state of a keypad button <btn> for the keypad at index/port, where <state> is CLICK or DOUBLECLICK
setButtonStatus(btnNum,bs)	IKeypadComponent	<btn>				Discrete Function Channel: Set the status of a keypad button <btn> for the keypad at index/port (see state chart)
setButtonStatus(btnNum,bs)	IKeypadComponent	<btn>+100				Discrete Function Channel: Set the status of a keypad button <btn> for the keypad at index/port (see state chart)
processButtonStateEvent (PUSH)	IKeypadComponentListener	<btn>				Input Function Channel: Button state is Released when BUTTON_EVENT-RELEASE is received, where <btn> is the button/channel number
processButtonStateEvent (RELEASE)	IKeypadComponentListener	<btn>				Input Function Channel: Button state is Pushed when BUTTON_EVENT-PUSH is received, where <btn> is the button/channel number
processButtonStateEvent (ButtonState)	IKeypadComponentListener			KEYPADBTN-<btn>,<state>		Button State changed. <btn> is button number and <state> is CLICK or DOUBLECLICK

Light Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processButtonStatusEvent (ON)	IKeypadComponentListener	<btn>				Feedback Channel: Button status on ON while channel <btn> is on and channel <btn>+100 is off, where <btn> is the button/channel number (see chart below)
processButtonStatusEvent (BLINK)	IKeypadComponentListener	<btn>+100				Feedback Channel: Button status on BLINK while channel <btn> is on and channel <btn>+100 is on, where <btn> is the button/channel number (see chart below)
addLightComponent (index,lightAddress)	ILightComponent			LIGHTADD-<index>, <lightAddress>		Add a light at a given index, where <index> is 1 through x and <address> is a light address and x is the maximum supported light index (see specific module documentation)
cycleLight(index)	ILightComponent			LIGHTSTATE-<index>, TOGGLE		Cycle the state of a light for the light at index <index>. This command is relevant for light loads and scenes.
cycleLight(1)	ILightComponent			LIGHTSTATE-1,TOGGLE		Toggle Light Preset 1
cycleLight(10)	ILightComponent			LIGHTSTATE-10,TOGGLE		Toggle Light Preset 10
cycleLight(11)	ILightComponent			LIGHTSTATE-11,TOGGLE		Toggle Light Preset 11
cycleLight(12)	ILightComponent			LIGHTSTATE-12,TOGGLE		Toggle Light Preset 12
cycleLight(13)	ILightComponent			LIGHTSTATE-13,TOGGLE		Toggle Light Preset 13
cycleLight(14)	ILightComponent			LIGHTSTATE-14,TOGGLE		Toggle Light Preset 14
cycleLight(15)	ILightComponent			LIGHTSTATE-15,TOGGLE		Toggle Light Preset 15
cycleLight(16)	ILightComponent			LIGHTSTATE-16,TOGGLE		Toggle Light Preset 16
cycleLight(17)	ILightComponent			LIGHTSTATE-17,TOGGLE		Toggle Light Preset 17
cycleLight(18)	ILightComponent			LIGHTSTATE-18,TOGGLE		Toggle Light Preset 18
cycleLight(19)	ILightComponent			LIGHTSTATE-19,TOGGLE		Toggle Light Preset 19
cycleLight(2)	ILightComponent			LIGHTSTATE-2,TOGGLE		Toggle Light Preset 2
cycleLight(20)	ILightComponent			LIGHTSTATE-20,TOGGLE		Toggle Light Preset 20
cycleLight(3)	ILightComponent			LIGHTSTATE-3,TOGGLE		Toggle Light Preset 3
cycleLight(4)	ILightComponent			LIGHTSTATE-4,TOGGLE		Toggle Light Preset 4
cycleLight(5)	ILightComponent			LIGHTSTATE-5,TOGGLE		Toggle Light Preset 5
cycleLight(6)	ILightComponent			LIGHTSTATE-6,TOGGLE		Toggle Light Preset 6
cycleLight(7)	ILightComponent			LIGHTSTATE-7,TOGGLE		Toggle Light Preset 7
cycleLight(8)	ILightComponent			LIGHTSTATE-8,TOGGLE		Toggle Light Preset 8
cycleLight(9)	ILightComponent			LIGHTSTATE-9,TOGGLE		Toggle Light Preset 9

Light Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getLightComponentAddress (index)	ILightComponent			?LIGHTADDR-<index>		Query for the address of the light at index <index>, responds with LIGHTADDR-<index>,<address>
getLightComponentIndex (lightAddress)	ILightComponent			?LIGHTIDX-<address>		Query for the index of the light with address <address>, responds with LIGHTADDR-<index>,<address>
getLightLevel(index)	ILightComponent			?LIGHTLEVEL-<index>		Query for the level of a light for the light at index <index>, responds with LIGHTLEVEL-<index>,<level> where <level> is 0-255. This command is relevant for light loads only.
isLightOn(index)	ILightComponent			?LIGHTSTATE-<index>		Query for the state of a light for the light at index <index>, responds with LIGHTSTATE-<index>,<state> where <state> is ON or OFF. This command is relevant for light loads, presets, and scenes.
isLightOn(1)	ILightComponent			?LIGHTSTATE-1		Query status of Light Preset 1
isLightOn(10)	ILightComponent			?LIGHTSTATE-10		Query status of Light Preset 10
isLightOn(11)	ILightComponent			?LIGHTSTATE-11		Query status of Light Preset 11
isLightOn(12)	ILightComponent			?LIGHTSTATE-12		Query status of Light Preset 12
isLightOn(13)	ILightComponent			?LIGHTSTATE-13		Query status of Light Preset 13
isLightOn(14)	ILightComponent			?LIGHTSTATE-14		Query status of Light Preset 14
isLightOn(15)	ILightComponent			?LIGHTSTATE-15		Query status of Light Preset 15
isLightOn(16)	ILightComponent			?LIGHTSTATE-16		Query status of Light Preset 16
isLightOn(17)	ILightComponent			?LIGHTSTATE-17		Query status of Light Preset 17
isLightOn(18)	ILightComponent			?LIGHTSTATE-18		Query status of Light Preset 18
isLightOn(19)	ILightComponent			?LIGHTSTATE-19		Query status of Light Preset 19
isLightOn(2)	ILightComponent			?LIGHTSTATE-2		Query status of Light Preset 2
isLightOn(20)	ILightComponent			?LIGHTSTATE-20		Query status of Light Preset 20
isLightOn(3)	ILightComponent			?LIGHTSTATE-3		Query status of Light Preset 3
isLightOn(4)	ILightComponent			?LIGHTSTATE-4		Query status of Light Preset 4
isLightOn(5)	ILightComponent			?LIGHTSTATE-5		Query status of Light Preset 5
isLightOn(6)	ILightComponent			?LIGHTSTATE-6		Query status of Light Preset 6
isLightOn(7)	ILightComponent			?LIGHTSTATE-7		Query status of Light Preset 7
isLightOn(8)	ILightComponent			?LIGHTSTATE-8		Query status of Light Preset 8
isLightOn(9)	ILightComponent			?LIGHTSTATE-9		Query status of Light Preset 9

Light Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
removeLightComponent (lightAddress)	ILightComponent			LIGHTREMOVEADDR-<lightAddress>		Remove the light with address <address>, where <address> is a light address
removeLightComponent (index)	ILightComponent			LIGHTREMOVEIDX-<index>		Remove the light at index <index>, where <index> is 1 through x and x is the maximum supported light index (see specific module documentation)
setLightLevel (index,level,seconds)	ILightComponent			LIGHTLEVEL-<index>, <level>,<seconds>		Set the level of a light for the light at index <index> in <time>, <level> where <level> is 0-255 and <time> is in seconds. This command is relevant for light loads only.
setLightOn(index,state)	ILightComponent			LIGHTSTATE-<index>,<state>		Set the state of a light for the light at index <index>, where <state> is ON or OFF. This command is relevant for light loads, presets, and scenes.
processLightEvent(int,boolean)	ILightComponentListener			LIGHTSTATE-<index>,<state>		State of a light changed for the light at index <index>, <state> is ON or OFF
processLightEvent(1,boolean)	ILightComponentListener			LIGHTSTATE-1,<state>		Response to ?LIGHTSTATE-1
processLightEvent(10,boolean)	ILightComponentListener			LIGHTSTATE-10,<state>		Response to ?LIGHTSTATE-10
processLightEvent(11,boolean)	ILightComponentListener			LIGHTSTATE-11,<state>		Response to ?LIGHTSTATE-11
processLightEvent(12,boolean)	ILightComponentListener			LIGHTSTATE-12,<state>		Response to ?LIGHTSTATE-12
processLightEvent(13,boolean)	ILightComponentListener			LIGHTSTATE-13,<state>		Response to ?LIGHTSTATE-13
processLightEvent(14,boolean)	ILightComponentListener			LIGHTSTATE-14,<state>		Response to ?LIGHTSTATE-14
processLightEvent(15,boolean)	ILightComponentListener			LIGHTSTATE-15,<state>		Response to ?LIGHTSTATE-15
processLightEvent(16,boolean)	ILightComponentListener			LIGHTSTATE-16,<state>		Response to ?LIGHTSTATE-16
processLightEvent(17,boolean)	ILightComponentListener			LIGHTSTATE-17,<state>		Response to ?LIGHTSTATE-17
processLightEvent(18,boolean)	ILightComponentListener			LIGHTSTATE-18,<state>		Response to ?LIGHTSTATE-18
processLightEvent(19,boolean)	ILightComponentListener			LIGHTSTATE-19,<state>		Response to ?LIGHTSTATE-19
processLightEvent(2,boolean)	ILightComponentListener			LIGHTSTATE-2,<state>		Response to ?LIGHTSTATE-2
processLightEvent(20,boolean)	ILightComponentListener			LIGHTSTATE-20,<state>		Response to ?LIGHTSTATE-20
processLightEvent(3,boolean)	ILightComponentListener			LIGHTSTATE-3,<state>		Response to ?LIGHTSTATE-3
processLightEvent(4,boolean)	ILightComponentListener			LIGHTSTATE-4,<state>		Response to ?LIGHTSTATE-4
processLightEvent(5,boolean)	ILightComponentListener			LIGHTSTATE-5,<state>		Response to ?LIGHTSTATE-5
processLightEvent(6,boolean)	ILightComponentListener			LIGHTSTATE-6,<state>		Response to ?LIGHTSTATE-6
processLightEvent(7,boolean)	ILightComponentListener			LIGHTSTATE-7,<state>		Response to ?LIGHTSTATE-7
processLightEvent(8,boolean)	ILightComponentListener			LIGHTSTATE-8,<state>		Response to ?LIGHTSTATE-8
processLightEvent(9,boolean)	ILightComponentListener			LIGHTSTATE-9,<state>		Response to ?LIGHTSTATE-9

Light Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processLightLevelEvent(int,int)	ILightComponentListener			LIGHTLEVEL-<index>,<level>		Level of a light changed for the light at index <index>, <level> where <level> is 0-255
processLightLevelRampEvent (int,LevelRamp)	ILightComponentListener			LIGHTRAMP-<index>,<state>		Light is ramping for the light at index <index>, <state> is UP, DOWN or STOP
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

Light State Charts

processButtonStateEvent	
State	**Channel <btn>**
RELEASE	OFF
PUSH	ON

processButtonStatusEvent		
State	**Channel <btn>**	**Channel <btn>+100**
OFF	OFF	OFF
ON	ON	OFF
BLINK	ON	ON

Note: *All Keypad commands include an index.*

This index is used to obtain the keypad component and the function is called on that component.

The range for channel '<btn>' is 1 to 100; the corresponding range for channel '<btn> + 100' is 101 to 200. Channels above 200 are passed to the module for advanced processing.

LightSystem

- IKeypadSystemComponent
- IKeypadSystemComponentListener
- ILightSystemComponent
- ILightSystemComponentListener
- IModuleComponent
- IModuleComponentListener

Name: LightSystem						
Interface: LightSystem						
LightSystem Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getKeypadSystemButtonState (keypadAddress)	IKeypadSystemComponent			?KEYPADSYSTEMBUTTONSTATE-<keypadAddress>		Query the state of a button, at the given button <address>. Responds with "KEYPADSYSTEMBUTTON-STATE-<state>", where <state> is CLICK, DOUBLECLICK, PUSH, or RELEASE.
getKeypadSystemButtonStatus (keypadAddress)	IKeypadSystemComponent			?KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>		Query the status of a button, at the given button <address>. Responds with "KEYPADSYSTEMBUTTON STATUS-<address>,<status>" where <status> is ON, OFF, or BLINK.
setKeypadSystemButtonState (keypadAddress,buttonState)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,<buttonState>		Set the state of a button, at the given button <address>, for the keypad at index/port, to a button state of CLICK, DOUBLECLICK, PUSH, or RELEASE.
setKeypadSystemButtonState (keypadAddress,CLICK)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,CLICK		Set the state of a button, at the given button <address>, for the keypad at index/port, to a button state of CLICK, DOUBLECLICK, PUSH, or RELEASE.
setKeypadSystemButtonState(keypadAddress,DOUBLECLICK)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,DOUBLECLICK		Set the state of a button, at the given button <address>, for the keypad at index/port, to a button state of CLICK, DOUBLECLICK, PUSH, or RELEASE.
setKeypadSystemButtonState (keypadAddress,PUSH)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,PUSH		Set the state of a button, at the given button <address>, for the keypad at index/port, to a button state of CLICK, DOUBLECLICK, PUSH, or RELEASE.

LightSystem Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setKeypadSystemButtonState (keypadAddress,RELEASE)	IIKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,RELEASE		Set the state of a button, at the given button <address>, for the keypad at index/port, to a button state of CLICK, DOUBLECLICK, PUSH, or RELEASE.
setKeypadSystemButtonStatus (keypadAddress,buttonStatus)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>,<buttonStatus>		Set the status of a button, at the given button <address>, for the keypad at index/port. to a button status of ON, OFF, or BLINK
setKeypadSystemButtonStatus (keypadAddress,BLINK)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>,BLINK		Set the status of a button, at the given button <address>, for the keypad at index/port. to a button status of ON, OFF, or BLINK
setKeypadSystemButtonStatus (keypadAddress,OFF)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>,OFF		Set the status of a button, at the given button <address>, for the keypad at index/port. to a button status of ON, OFF, or BLINK
setKeypadSystemButtonStatus (keypadAddress,ON)	IKeypadSystemComponent			KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>,ON		Set the status of a button, at the given button <address>, for the keypad at index/port. to a button status of ON, OFF, or BLINK
processKeypadSystemButtonStateEvent (keypadAddress,CLICK)	IKeypadSystemComponentListener			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,<buttonState>		State of a button changed at the given button <address>, where <state> is CLICK, DOUBLECLICK, PUSH, or RELEASE.
processKeypadSystemButtonStateEvent (keypadAddress,DOUBLECLICK)	IKeypadSystemComponentListener			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,<buttonState>		State of a button changed at the given button <address>, where <state> is CLICK, DOUBLECLICK, PUSH, or RELEASE.
processKeypadSystemButtonStateEvent (keypadAddress,PUSH)	IKeypadSystemComponentListener			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,<buttonState>		State of a button changed at the given button <address>, where <state> is CLICK, DOUBLECLICK, PUSH, or RELEASE.
processKeypadSystemButtonStateEvent (keypadAddress,RELEASE)	IKeypadSystemComponentListener			KEYPADSYSTEMBUTTONSTATE-<keypadAddress>,<buttonState>		State of a button changed at the given button <address>, where <state> is CLICK, DOUBLECLICK, PUSH, or RELEASE.
processKeypadSystemButtonStatusEvent (keypadAddress,BLINK)	IKeypadSystemComponentListener			KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>,<buttonStatus>		Status of a button changed at the given button <address>, where <status> is ON, OFF, or BLINK.
processKeypadSystemButtonStatusEvent (keypadAddress,OFF)	IKeypadSystemComponentListener			KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>,<buttonStatus>		Status of a button changed at the given button <address>, where <status> is ON, OFF, or BLINK.

LightSystem Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processKeypadSystemButtonStatusEvent (keypadAddress,ON)	IKeypadSystemComponentListener			KEYPADSYSTEMBUTTONSTATUS-<keypadAddress>,<buttonStatus>		Status of a button changed at the given button <address>, where <status> is ON, OFF, or BLINK.
cycleLight(address)	ILightSystemComponent			LIGHTSYSTEMSTATE-<address>, TOGGLE		Cycle the state of a light, at the given light <address>. This command is relevant for light loads and scenes.
getLightLevel(address)	ILightSystemComponent			?LIGHTSYSTEMLEVEL-<address>		Query the level of a light, at the given light <address>. Responds with "LIGHTSYSTEMLEVEL-<address>,<level>", where <level> is 0-255. This command is relevant for light loads only.
isLightOn(address)	ILightSystemComponent			?LIGHTSYSTEMSTATE-<address>		Query the state of a light, at the given light <address>. Responds with "LIGHTSYSTEMSTATE-<address>,<state>" where <state> is true for ON, or false for OFF. This command is relevant for light loads, presets, and scenes.
setLight(address,state)	ILightSystemComponent			LIGHTSYSTEMSTATE-<address>, <lightstate>		Set the state of a light, at the given light <address>, where <lightstate> is ON or OFF. This command is relevant for light loads, presets, and scenes.
setLightLevel(address)	ILightSystemComponent			?LIGHTSYSTEMLEVEL-<address>		Query the level of a light, at the given light <address>. Responds with "LIGHTSYSTEMLEVEL-<address>,<level>", where <level> is 0-255. This command is relevant for light loads only.
setLightLevel(address,level)	ILightSystemComponent			LIGHTSYSTEMLEVEL-<address>, <level>		Set the level of a light, at the given light <address>, to a given <level> where <level> is 0-255. This command is relevant for light loads only.
setLightLevel(address,level,time)	ILightSystemComponent			LIGHTSYSTEMLEVEL-<address>, <level>,<time>		Set the level of a light, at the given light <address>, to a given <level>, ramped over the specified <time>, where <level> is 0-255 and <time> is in seconds. This command is relevant for light loads only.
setLightLevelRamp(address,DOWN)	ILightSystemComponent			LIGHTSYSTEMRAMP-<address>, DOWN		Ramp Down a light level, at the given light <address>, until LIGHTSYSTEMRAMP-<address>,STOP is sent. This command is relevant for light loads and scenes.

LightSystem Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setLightLevelRamp(address,STOP)	ILightSystemComponent			LIGHTSYSTEMRAMP-<address>, STOP		Stop ramping the light level, at the given light <address>. This command is relevant for light loads and scenes.
setLightLevelRamp(address,UP)	ILightSystemComponent			LIGHTSYSTEMRAMP-<address>,UP		Ramp Up a light level, at the given light <address>, until LIGHTSYSTEMRAMP-<address>,STOP is sent. This command is relevant for light loads and scenes.
setLightOff(address)	ILightSystemComponent			LIGHTSYSTEMSTATE-<address>, OFF		Set the state of a light OFF, at the given light <address>. This is implemented in base and should not be overridden. This command is relevant for light loads, presets, and scenes.
setLightOn(address)	ILightSystemComponent			LIGHTSYSTEMSTATE-<address>, ON		Set the state of a light ON, at the given light <address>. This is implemented in base and should not be overridden. This command is relevant for light loads, presets, and scenes.
processLightLevelEvent(address,level)	ILightSystemComponentListener			LIGHTSYSTEMLEVEL-<address>, <level>		Level of a light changed at the given <address>, where <level> is an integer from 0-255.
processLightLevelRampEvent (address,DOWN)	ILightSystemComponentListener			LIGHTSYSTEMRAMP-<address>, <control>		Light is ramping at the given <address>, where ramp <control> is UP, DOWN, or STOP.
processLightLevelRampEvent (address,STOP)	ILightSystemComponentListener			LIGHTSYSTEMRAMP-<address>, <control>		Light is ramping at the given <address>, where ramp <control> is UP, DOWN, or STOP.
processLightLevelRampEvent (address,UP)	ILightSystemComponentListener			LIGHTSYSTEMRAMP-<address>, <control>		Light is ramping at the given <address>, where ramp <control> is UP, DOWN, or STOP.
processLightStateEvent(address,state)	ILightSystemComponentListener			LIGHTSYSTEMSTATE-<address>, <state>		State of a light changed at the given <address>, where <state> is ON or OFF.
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>

LightSystem Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

Monitor

- IDisplayComponent
- IDisplayComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IMonitorComponent
- IMonitorComponentListener
- IPowerComponent
- IPowerComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Monitor						
Interface: IMonitor						
Monitor Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
adjustBrightness(1)	IDisplayComponent	148			BRIGHT_UP	Momentary Function Channel: Brightness is incremented when channel is activated
adjustBrightness(-1)	IDisplayComponent	149			BRIGHT_DN	Momentary Function Channel: Brightness is decremented when channel is activated
adjustColor(1)	IDisplayComponent	150			COLOR_UP	Momentary Function Channel: Color is incremented when channel is activated
adjustColor(-1)	IDisplayComponent	151			COLOR_DN	Momentary Function Channel: Color is decremented when channel is activated
adjustContrast(1)	IDisplayComponent	152			CONTRAST_UP	Momentary Function Channel: Contrast is incremented when channel is activated
adjustContrast(-1)	IDisplayComponent	153			CONTRAST_DN	Momentary Function Channel: Contrast is decremented when channel is activated
adjustSharpness(1)	IDisplayComponent	154			SHARP_UP	Momentary Function Channel: Sharpness is incremented when channel is activated
adjustSharpness(-1)	IDisplayComponent	155			SHARP_DN	Momentary Function Channel: Sharpness is decremented when channel is activated
adjustTint(1)	IDisplayComponent	156			TINT_UP	Momentary Function Channel: Tint is incremented when channel is activated
adjustTint(-1)	IDisplayComponent	157			TINT_DN	Momentary Function Channel: Tint is decremented when channel is activated

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleAspectRatio()	IDisplayComponent	142			ASPECT_RATIO	Momentary Function Channel: Cycle aspect ratios when channel is activated
cycleFreeze()	IDisplayComponent	213			PIC_FREEZE	Momentary Function Channel: Cycle freeze when channel is activated
cyclePictureMute()	IDisplayComponent	210			PIC_MUTE	Momentary Function Channel: Cycle picture mute when channel is activated
cyclePIP()	IDisplayComponent	194			PIP	Momentary Function Channel: Cycle PIP when channel is activated
cyclePIPPosition()	IDisplayComponent	191			PIP_POS	Momentary Function Channel: Cycle PIP positions when channel is activated
getActiveWindow()	IDisplayComponent			?ACTIVEWINDOW		Query active window, responds with ACTIVEWINDOW-<window>, where <window> is LEFT,RIGHT,MAIN,SUB
getAspectRatio()	IDisplayComponent			?ASPECT		Query aspect ratio, responds with ASPECT-<ratio>, where <ratio> is ANAMORPHIC, WIDESCREEN,NORMAL
getAspectRatioCount()	IDisplayComponent			?ASPECTRATIOCOUNT		Query aspect ratio count, responds with ASPECTRATIOCOUNT-<count>
getAspectRatioProperties(index)	IDisplayComponent			?ASPECTRATIOPROPERTIES		Query properties for all aspect ratios, responds with ASPECTRATIOPROPERTIES-<index>, <displayName>,<value>[;<index>, <displayName>,<value>]*
getAspectRatioProperty(index)	IDisplayComponent			?ASPECTRATIOPROPERTY-<index>		Query properties for single aspect ratio, responds with ASPECTRATIOPROPERTY-<index>, <displayName>,<value>
getAspectRatioSelect()	IDisplayComponent			?ASPECTRATIOSELECT		Gets the index of the currently selected aspect ratio property.
getVideoType()	IDisplayComponent			?VIDEOTYPE		Query video type, responds with VIDEOTYPE-<type> where <type> is AUTO,NTSC,PAL, SECAM
setActiveWindow(mss)	IDisplayComponent			ACTIVEWINDOW-<mss>		Set active window, where <mss> is LEFT,RIGHT, MAIN,SUB
setAspectRatio(aspectRatio)	IDisplayComponent			ASPECT-<aspectRatio>		Set aspect ratio, where <aspectRatio> is ANAMORPHIC,WIDESCREEN,NORMAL
setAspectRatioSelect(index)	IDisplayComponent			ASPECTRATIOSELECT-<index>		Sets the current aspect ratio, where <index> is an integer number between 1 and the value returned by ?ASPECTRATIOSELECT, responds with ASPECTRATIOSELECT-<index>
setBrightness(level)	IDisplayComponent		10		BRIGHT_LVL	Set brightness level, range is 0-255

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setColor(level)	IDisplayComponent		11		COLOR_LVL	Set color level, range is 0-255
setContrast(level)	IDisplayComponent		12		CONTRAST_LVL	Set contrast level, range is 0-255
setFreezeOn(state)	IDisplayComponent	214			PIC_FREEZE_ON	Discrete Function Channel: Freeze is on while channel is active
setPictureMuteOn(state)	IDisplayComponent	211			PIC_MUTE_ON	Discrete Function Channel: Picture Mute is on while channel is active
setPIPOn(state)	IDisplayComponent	195			PIP_ON	Discrete Function Channel: PIP is on while channel is active
setSharpness(level)	IDisplayComponent		13		SHARP_LVL	Set sharpness level, range is 0-255
setTint(level)	IDisplayComponent		14		TINT_LVL	Set tint level, range is 0-255
setVideoType(vt)	IDisplayComponent			VIDEOTYPE-<vt>		Set video type, where <vt> is AUTO,NTSC,PAL,SECAM
swapPIP()	IDisplayComponent	193			PIP_SWAP	Momentary Function Channel: Swap PIP when channel is activated
processActiveWindowEvent (MultiScreenSelect)	IDisplayComponentListener			ACTIVEWINDOW-<window>		Active window changed, where <window> is LEFT,RIGHT,MAIN,SUB
processAspectRatioCountEvent(count)	IDisplayComponentListener			ASPECTRATIOCOUNT-<count>		Responds with aspect ratio count, where <count> is an integer value
processAspectRatioEvent (AspectRatio)	IDisplayComponentListener			ASPECT-<ratio>		Aspect ratio changed, where <ratio> is ANAMORPHIC,WIDESCREEN,NORMAL
processAspectRatioPropertiesEvent (String[])	IDisplayComponentListener			ASPECTRATIOPROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>,<value>,...]		Returns properties for each supported aspect ratio, where <index> is an integer, <displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioPropertyEvent (String[])	IDisplayComponentListener			ASPECTRATIOPROPERTY-<index>,<displayName>, <value>		Returns properties for single aspect ratio, where <index> is an integer,<displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioSelectEvent (index)	IDisplayComponentListener			ASPECTRATIOSELECT-<index>		Returns <index> of currently selected aspect ratio.
processBrightnessEvent(int)	IDisplayComponentListener		10		BRIGHT_LVL	Brightness changed, range is 0-255
processColorEvent(int)	IDisplayComponentListener		11		COLOR_LVL	Color changed, range is 0-255
processContrastEvent(int)	IDisplayComponentListener		12		CONTRAST_LVL	Contrast changed, range is 0-255
processFreezeEvent(boolean)	IDisplayComponentListener	214			PIC_FREEZE_FB	Feedback Channel: Freeze is on if channel is on
processPictureMuteEvent(boolean)	IDisplayComponentListener	211			PIC_MUTE_FB	Feedback Channel: Picture is muted if channel is on
processPIPEvent(boolean)	IDisplayComponentListener	195			PIP_FB	Feedback Channel: PIP is on if channel is on

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processSharpnessEvent(int)	IDisplayComponentListener		13		SHARP_LVL	Sharpness changed, range is 0-255
processTintEvent(int)	IDisplayComponentListener		14		TINT_LVL	Tint changed, range is 0-255
processVideoTypeEvent(VideoType)	IDisplayComponentListener			VIDEOTYPE-<type>		Video type changed, where <type> is AUTO,NTSC,PAL,SECAM
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton (THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton (UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton (PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton (INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processInputPropertiesEvent (String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputPropertyEvent (String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent (boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on

Monitor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Monitor State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Motor

- IModuleComponent
- IModuleComponentListener
- IMotorComponent
- IMotorComponentListener

Name: Motor						
Interface: IMotor						
Motor Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

Motor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleMotorPreset()	IMotorComponent	187			MOTOR_PRESET	Momentary Function Channel: Cycle Motor preset when channel is activated
getMotorPreset()	IMotorComponent			?MOTORPRESET		Query for Motor preset, responds with MOTORPRESET-<preset>
setMotorDirection(STOP)	IMotorComponent	2			MOTOR_STOP	Momentary Function Channel: Set Motor direction to stop, causing motor to stop between opened and closed
setMotorDirection(OPEN)	IMotorComponent	4			MOTOR_OPEN	Momentary Function Channel: Set Motor direction to open, causing motor to move in the OPEN direction
setMotorDirection(CLOSE)	IMotorComponent	5			MOTOR_CLOSE	Momentary Function Channel: Set Motor direction to close, causing motor to move in the CLOSE direction
setMotorPosition(position)	IMotorComponent		6		MOTOR_POS_LVL	Recall Motor position, range is 0-255, 0 is close, 255 is open
setMotorPreset(preset)	IMotorComponent			MOTORPRESET-<preset>		Set Motor preset where <preset> is 1 through x where x is the maximum supported preset (see specific module documentation)
processMotorDirectionEvent (STOP)	IMotorComponentListener	2			MOTOR_STOP_FB	Feedback Channel: Motor is stopped between opened and closed
processMotorDirectionEvent (OPEN)	IMotorComponentListener	4			MOTOR_OPEN_FB	Feedback Channel: Motor direction is open, motor is moving to the OPEN position or is open
processMotorDirectionEvent (CLOSE)	IMotorComponentListener	5			MOTOR_CLOSE_FB	Feedback Channel: Motor direction is close, motor is moving to the CLOSE position or is closed
processMotorPositionEvent(int)	IMotorComponentListener		6		MOTOR_POS_LVL	Motor position changed, range is 0-255, 0 is close, 255 is open
processMotorPresetEvent(int)	IMotorComponentListener			MOTORPRESET-<preset>		Motor preset changed where <preset> is 1 through x where x is the maximum supported preset (see specific module documentation)

Motor State Charts

processMotorDirectionEvent			
State	**Channel 2**	**Channel 4**	**Channel 5**
STOP	ON	OFF	OFF
OPEN	OFF	ON	OFF
CLOSE	OFF	OFF	ON

Multi Window

- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IMultiWindowComponent
- IMultiWindowComponentListener
- IPowerComponent
- IPowerComponentListener
- IWindowComponent
- IWindowComponentListener

Name: Multi Window						
Interface: IMultiWindow						
Multi Window Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied file-name. Used to load meta-data properties for mod-ules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cycleMultiWindowPreset()	IMultiWindowComponent	136			MULTIWIN_PRESET	Momentary Function Channel: Cycle multi-window preset when channel is activated
getMultiWindowPreset()	IMultiWindowComponent			?MULTIWINPRESET		Query for multi-window preset, responds with MULTIWINPRESET-<preset>
saveMultiWindowPreset (nPresetNum)	IMultiWindowComponent			MULTIWINPRESETSAVE-<preset>		Save multi-window Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setMultiWindowPreset (nPresetNum)	IMultiWindowComponent			MULTIWINPRESET-<preset>		Recall multi-window preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
processMultiWindowPresetEvent(int)	IMultiWindowComponentListener			MULTIWINPRESET-<preset>		Multi-window preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
addWindowComponent (index,windowAddress)	IWindowComponent			WINDOWADD-<index>, <windowAddress>		Add a window at a given index, where <index> is 1 through x and <address> is a window address and x is the maximum supported window index (see specific module documentation)
adjustBrightness(1)	IWindowComponent	148			BRIGHT_UP	Momentary Function Channel: Brightness is incremented when channel is activated
adjustBrightness(-1)	IWindowComponent	149			BRIGHT_DN	Momentary Function Channel: Brightness is decremented when channel is activated
adjustColor(1)	IWindowComponent	150			COLOR_UP	Momentary Function Channel: Color is incremented when channel is activated
adjustColor(-1)	IWindowComponent	151			COLOR_DN	Momentary Function Channel: Color is decremented when channel is activated
adjustContrast(1)	IWindowComponent	152			CONTRAST_UP	Momentary Function Channel: Contrast is incremented when channel is activated
adjustContrast(-1)	IWindowComponent	153			CONTRAST_DN	Momentary Function Channel: Contrast is decremented when channel is activated
adjustSharpness(1)	IWindowComponent	154			SHARP_UP	Momentary Function Channel: Sharpness is incremented when channel is activated
adjustSharpness(-1)	IWindowComponent	155			SHARP_DN	Momentary Function Channel: Sharpness is decremented when channel is activated
adjustTint(1)	IWindowComponent	156			TINT_UP	Momentary Function Channel: Tint is incremented when channel is activated
adjustTint(-1)	IWindowComponent	157			TINT_DN	Momentary Function Channel: Tint is decremented when channel is activated
cycleFreeze()	IWindowComponent	213			PIC_FREEZE	Momentary Function Channel: Cycle freeze when channel is activated
cyclePictureMute()	IWindowComponent	210			PIC_MUTE	Momentary Function Channel: Cycle picture mute when channel is activated
getImagePosition()	IWindowComponent			?IMAGE		Query for the image position. Response with IMAGE-<source>,<x>,<y>,<height>,<width> where <source> is 1 to the maximum source number (see specific module documentation)
getImageSize()	IWindowComponent			?IMAGE		Query for the image size. Response with IMAGE-<source>,<x>,<y>,<height>,<width> where <source> is 1 to the maximum source number (see specific module documentation)
getImageSource()	IWindowComponent			?IMAGE		Query for the image source. Response with IMAGE-<source>,<x>,<y>,<height>,<width> where <source> is 1 to the maximum source number (see specific module documentation)

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getWindowComponentAddress (index)	IWindowComponent			?WINDOWADDR-<index>		Query for the address of the window at index <index>, responds with WINDOWADDR-<index>, <address>
getWindowComponentIndex (windowAddress)	IWindowComponent			?WINDOWIDX-<address>		Query for the index of the window with address <address>, responds with WINDOWADDR-<index>, <address>
getWindowPosition()	IWindowComponent			?WINDOW		Query for the window position. Response with WINDOW-<x>,<y>,<height>,<width>
getWindowSize()	IWindowComponent			?WINDOW		Query for the window size. Response with WINDOW-<x>,<y>,<height>,<width>
getZOrder()	IWindowComponent			?WINDOWZORDER		Get the Z-order. Responds with WINDOWZORDER-<value> where <value> is 1 to the maximum z-order of the device, 1 is the top-most z-order (see specific module documentation)
pan(UP)	IWindowComponent	132			PAN_UP	Momentary Function Channel: Image is moved up one step within the window when channel is activated
pan(DOWN)	IWindowComponent	133			PAN_DN	Momentary Function Channel: Image is moved down one step within the window when channel is activated
pan(LEFT)	IWindowComponent	134			PAN_LT	Momentary Function Channel: Image is moved left one step within the window when channel is activated
pan(RIGHT)	IWindowComponent	135			PAN_RT	Momentary Function Channel: Image is moved right one step within the window when channel is activated
removeWindowComponent (windowAddress)	IWindowComponent			WINDOWREMOVEADDR-<windowAddress>		Remove the window with address <address>, where <address> is a window address
removeWindowComponent(index)	IWindowComponent			WINDOWREMOVEIDX-<index>		Remove the window at index <index>, where <index> is 1 through x and x is the maximum supported window index (see specific module documentation)
setBrightness(level)	IWindowComponent		10		BRIGHT_LVL	Set brightness level, range is 0-255
setColor(level)	IWindowComponent		11		COLOR_LVL	Set color level, range is 0-255
setContrast(level)	IWindowComponent		12		CONTRAST_LVL	Set contrast level, range is 0-255
setFreezeOn(state)	IWindowComponent	214			PIC_FREEZE_ON	Discrete Function Channel: Freeze is on while channel is active

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setImage(source,pos,size)	IWindowComponent			IMAGE-<source>,<x>,<y>,<height>,<width>		Set the image source, position and size relative to the window where <source> is 1 to the maximum source number (see specific module documentation) and <x> and <y> are the window coordinates for the upper-left hand corner of the image and <height> and <width> are the size of the image to be displayed in the window. Used to change the position of the image visible in the window.
setImageSource(int)	IWindowComponent			IMAGESOURCE-<source>		Set the image source for the window where <source> is 1 to the maximum source number (see specific module documentation)
setPictureMuteOn(state)	IWindowComponent	211			PIC_MUTE_ON	Discrete Function Channel: Picture Mute is on while channel is active
setSharpness(level)	IWindowComponent		13		SHARP_LVL	Set sharpness level, range is 0-255
setTint(level)	IWindowComponent		14		TINT_LVL	Set tint level, range is 0-255
setWindow(pos,size)	IWindowComponent			WINDOW-<x>,<y>,<height>,<width>		Set the window position and size relative to the display where <x> and <y> are the display coordinates for the upper-left hand corner of the window and <height> and <width> are the size of the window to be displayed in the display. Used to change the position of the window visible in the display.
setZOrder(position)	IWindowComponent			WINDOWZORDER-<position>		Shuffle the window z-order relative to the other windows where <position> is FRONT, BACK, FORWARD (up one level), BACKWARD (down one level)
setZOrder(value)	IWindowComponent			WINDOWZORDER-<value>		Set the Z-order where <value> is 1 to the maximum z-order of the device, 1 is the top-most z-order (see specific module documentation)
zoomIn()	IWindowComponent	159			ZOOM_IN	Momentary Function Channel: Image size is enlarged within the window (zoom in/tele) when channel is activated
zoomOut()	IWindowComponent	158			ZOOM_OUT	Momentary Function Channel: Image size is reduced within the window (zoom out/window) when channel is activated
processBrightnessEvent(int)	IWindowComponentListener		10		BRIGHT_LVL	Brightness changed, range is 0-255
processColorEvent(int)	IWindowComponentListener		11		COLOR_LVL	Color changed, range is 0-255
processContrastEvent(int)	IWindowComponentListener		12		CONTRAST_LVL	Contrast changed, range is 0-255
processFreezeEvent(boolean)	IWindowComponentListener	214			PIC_FREEZE_FB	Feedback Channel: Freeze is on if channel is on
processImageEvent(Position,Size)	IWindowComponentListener			IMAGE-<source>,<x>,<y>,<height>,<width>		Image position and/or size changed where <source> is 1 to the maximum source number (see specific module documentation)
processImageSourceEvent(int)	IWindowComponentListener			IMAGESOURCE-<source>		Image source for the window changed where <source> is 1 to the maximum source number (see specific module documentation)

Multi Window Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPictureMuteEvent(boolean)	IWindowComponentListener	211			PIC_MUTE_FB	Feedback Channel: Picture is muted if channel is on
processSharpnessEvent(int)	IWindowComponentListener		13		SHARP_LVL	Sharpness changed, range is 0-255
processTintEvent(int)	IWindowComponentListener		14		TINT_LVL	Tint changed, range is 0-255
processWindowEvent(Position,Size)	IWindowComponentListener			WINDOW-<x>,<y>, <height>,<width>		Window position and/or size changed.
processZOrderEvent(int)	IWindowComponentListener			WINDOWZORDER-<value>		Z-order changed where <value> is 1 to the maximum z-order of the device, 1 is the top-most z-order (see specific module documentation)

Multi Window State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Pool Spa

- IModuleComponent
- IModuleComponentListener
- IPoolSpaComponent
- IPoolSpaComponentListener

Name: Pool Spa						
Interface: IPoolSpa						
Pool Spa Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on

Pool Spa Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePoolHeatState()	IPoolSpaComponent	123			POOL_HEAT	Momentary Function Channel: Cycle Pool heat state when channel is activated
cycleSpaHeatState()	IPoolSpaComponent	124			SPA_HEAT	Momentary Function Channel: Cycle Spa heat state when channel is activated
cycleSpaJets()	IPoolSpaComponent	125			SPA_JETS	Momentary Function Channel: Cycle Spa jets when channel is activated
decrementPoolSetpoint()	IPoolSpaComponent	153			POOL_HEAT_DN	Momentary Function Channel: Pool setpoint is decremented when channel is activated
decrementSpaSetpoint()	IPoolSpaComponent	155			SPA_HEAT_DN	Momentary Function Channel: Spa setpoint is decremented when channel is activated
getPoolSpaTemperatureScale()	IPoolSpaComponent			?POOLSCALE		Query for the Pool/Spa temperature scale, responds with POOLSCALE-<scale> where <scale> is FAHRENHEIT, CELSIUS
incrementPoolSetpoint()	IPoolSpaComponent	152			POOL_HEAT_UP	Momentary Function Channel: Pool setpoint is incremented when channel is activated
incrementSpaSetpoint()	IPoolSpaComponent	154			SPA_HEAT_UP	Momentary Function Channel: Spa setpoint is incremented when channel is activated
isPoolSpaAuxOn(aux)	IPoolSpaComponent			?POOLAUX-<auxNumber>		Query for Pool/Spa Aux state, responds with POOLAUX-<aux number>,<state> where <state> is 0 (false) or 1 (true)
setPoolHeatState(OFF)	IPoolSpaComponent	174			POOL_HEAT_OFF	Momentary Function Channel: Set Pool heat state to off
setPoolHeatState(HEATER)	IPoolSpaComponent	175			POOL_HEATER	Momentary Function Channel: Set Pool heat state to heater
setPoolHeatState(SOLAR)	IPoolSpaComponent	176			POOL_SOLAR	Momentary Function Channel: Set Pool heat state to solar
setPoolHeatState(SOLAR_PREFERRED)	IPoolSpaComponent	177			POOL_SOLAR_PREF	Momentary Function Channel: Set Pool heat state to solar preferred
setPoolLightOn(state)	IPoolSpaComponent	172			POOL_LIGHT_ON	Discrete Function Channel: Pool light is on while channel is active
setPoolPumpOn(state)	IPoolSpaComponent	170			POOL_PUMP_ON	Discrete Function Channel: Pool pump is on while channel is active
setPoolSetpoint(nTemperature)	IPoolSpaComponent		39		POOL_HEAT_LVL	Set Pool setpoint, value is in degrees C or F depending on temperature scale

Pool Spa Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPoolSpaAuxOn(aux,state)	IPoolSpaComponent			POOLAUX-<auxNumber>,<state>		Set Pool/Spa Aux state where <aux number> is the number of the aux relay 1 to x where x is the maximum supported aux relay and <state> is 1 (on) or 0 (off) (see specific module documentation)
setPoolSpaTemperatureScale(ts)	IPoolSpaComponent			POOLSCALE-<ts>		Set the Pool/Spa temperature scale, where <ts> is FAHRENHEIT, CELSIUS
setSpaBlowerOn(state)	IPoolSpaComponent	186			SPA_BLOWER_ON	Discrete Function Channel: Spa blower is on while channel is active
setSpaHeatState(OFF)	IPoolSpaComponent	178			SPA_HEAT_OFF	Momentary Function Channel: Set Spa heat state to off
setSpaHeatState(HEATER)	IPoolSpaComponent	179			SPA_HEATER	Momentary Function Channel: Set Spa heat state to heater
setSpaHeatState(SOLAR)	IPoolSpaComponent	180			SPA_SOLAR	Momentary Function Channel: Set Spa heat state to solar
setSpaHeatState(SOLAR_PREFERRED)	IPoolSpaComponent	181			SPA_SOLAR_PREF	Momentary Function Channel: Set Spa heat state to solar preferred
setSpaJets(OFF)	IPoolSpaComponent	182			SPA_JETS_OFF	Momentary Function Channel: Set Spa jets to off
setSpaJets(LOW)	IPoolSpaComponent	183			SPA_JETS_LO	Momentary Function Channel: Set Spa jets to low
setSpaJets(MEDIUM)	IPoolSpaComponent	184			SPA_JETS_MED	Momentary Function Channel: Set Spa jets to medium
setSpaJets(HIGH)	IPoolSpaComponent	185			SPA_JETS_HI	Momentary Function Channel: Set Spa jets to high
setSpaLightOn(state)	IPoolSpaComponent	173			SPA_LIGHT_ON	Discrete Function Channel: Spa light is on while channel is active
setSpaPumpOn(state)	IPoolSpaComponent	171			SPA_PUMP_ON	Discrete Function Channel: Spa pump is on while channel is active
setSpaSetpoint(nTemperature)	IPoolSpaComponent		40		SPA_HEAT_LVL	Set Spa setpoint, value is in degrees C or F depending on temperature scale
processPoolHeatStateEvent(OFF)	IPoolSpaComponentListener	174			POOL_HEAT_OFF_FB	Feedback Channel: Pool heater set to off state, see state chart
processPoolHeatStateEvent(HEATER)	IPoolSpaComponentListener	175			POOL_HEATER_FB	Feedback Channel: Pool heater set to heat state, see state chart
processPoolHeatStateEvent(SOLAR)	IPoolSpaComponentListener	176			POOL_SOLAR_FB	Feedback Channel: Pool heater set to solar state, see state chart
processPoolHeatStateEvent (SOLAR_PREFERRED)	IPoolSpaComponentListener	177			POOL_SOLAR_PREF_FB	Feedback Channel: Pool heater set to solar preferred state, see state chart
processPoolHeatStatusEvent(HEATER)	IPoolSpaComponentListener	187			POOL_HEATING	Feedback Channel: Pool heater status is heater, see state chart
processPoolHeatStatusEvent(SOLAR)	IPoolSpaComponentListener	188			POOL_HEATING_SOLAR	Feedback Channel: Pool heater status is solar, see state chart

Pool Spa Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPoolLightOnEvent(boolean)	IPoolSpaComponentListener	172			POOL_LIGHT_FB	Feedback Channel: Pool light is on while channel is active
processPoolPumpOnEvent(boolean)	IPoolSpaComponentListener	170			POOL_PUMP_FB	Feedback Channel: Pool pump is on while channel is active
processPoolSetpointEvent(int)	IPoolSpaComponentListener		39		POOL_HEAT_LVL	Pool setpoint changed, value is in degrees C or F depending on temperature scale
processPoolSpaAuxOnEvent(int,boolean)	IPoolSpaComponentListener			POOLAUX-<auxNumber>,<state>		Pool/Spa Aux state changed where <aux number> is the number of the aux relay 1 to x where x is the maximum supported aux relay and <state> is 1 (on) or 0 (off) (see specific module documentation)
processPoolSpaOutdoorTemperatureEvent (float)	IPoolSpaComponentListener		34		OUTDOOR_TEMP_LVL	Outdoor air temperature changed, value is in degrees C or F depending on temperature scale
processPoolSpaTemperatureScaleEvent (TemperatureScale)	IPoolSpaComponentListener			POOLSCALE-<scale>		Pool temperature scale changed, <scale> is FAHRENHEIT,CELSIUS
processPoolTemperatureEvent(float)	IPoolSpaComponentListener		41		POOL_TEMP_LVL	Pool temperature changed, value is in degrees C or F depending on temperature scale
processSpaBlowerOnEvent(boolean)	IPoolSpaComponentListener	186			SPA_BLOWER_FB	Feedback Channel: Spa blower is on while channel is active
processSpaHeatStateEvent(OFF)	IPoolSpaComponentListener	178			SPA_HEAT_OFF_FB	Feedback Channel: Spa heater set to off state, see state chart
processSpaHeatStateEvent(HEATER)	IPoolSpaComponentListener	179			SPA_HEATER_FB	Feedback Channel: Spa heater set to heat state, see state chart
processSpaHeatStateEvent(SOLAR)	IPoolSpaComponentListener	180			SPA_SOLAR_FB	Feedback Channel: Spa heater set to solar state, see state chart
processSpaHeatStateEvent (SOLAR_PREFERRED)	IPoolSpaComponentListener	181			SPA_SOLAR_PREF_FB	Feedback Channel: Spa heater set to solar preferred state, see state chart
processSpaHeatStatusEvent(HEATER)	IPoolSpaComponentListener	189			SPA_HEATING	Feedback Channel: Spa heater status is heater, see state chart
processSpaHeatStatusEvent(SOLAR)	IPoolSpaComponentListener	190			SPA_HEATING_SOLAR	Feedback Channel: Spa heater status is solar, see state chart
processSpaJetsEvent(OFF)	IPoolSpaComponentListener	182			SPA_JETS_OFF_FB	Feedback Channel: Spa jets state is off, see state chart
processSpaJetsEvent(LOW)	IPoolSpaComponentListener	183			SPA_JETS_LO_FB	Feedback Channel: Spa jets state is low, see state chart
processSpaJetsEvent(MEDIUM)	IPoolSpaComponentListener	184			SPA_JETS_MED_FB	Feedback Channel: Spa jets state is medium, see state chart
processSpaJetsEvent(HIGH)	IPoolSpaComponentListener	185			SPA_JETS_HI_FB	Feedback Channel: Spa jets state is high, see state chart
processSpaLightOnEvent(boolean)	IPoolSpaComponentListener	173			SPA_LIGHT_FB	Feedback Channel: Spa light is on while channel is active

Pool Spa Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processSpaPumpOnEvent(boolean)	IPoolSpaComponentListener	171			SPA_PUMP_FB	Feedback Channel: Spa pump is on while channel is active
processSpaSetpointEvent(int)	IPoolSpaComponentListener		40		SPA_HEAT_LVL	Spa setpoint changed, value is in degrees C or F depending on temperature scale
processSpaTemperatureEvent(float)	IPoolSpaComponentListener		42		SPA_TEMP_LVL	Spa temperature changed, value is in degrees C or F depending on temperature scale

Pool Spa State Charts

processPoolHeatStateEvent				
State	**Channel 174**	**Channel 175**	**Channel 176**	**Channel 177**
OFF	ON	OFF	OFF	OFF
HEATER	OFF	ON	OFF	OFF
SOLAR	OFF	OFF	ON	OFF
SOLAR_PREFERRED	OFF	OFF	OFF	ON

processPoolHeatStatusEvent		
State	**Channel 187**	**Channel 188**
OFF	OFF	OFF
HEATER	ON	OFF
SOLAR	OFF	ON

processSpaHeatStateEvent				
State	**Channel 178**	**Channel 179**	**Channel 180**	**Channel 181**
OFF	ON	OFF	OFF	OFF
HEATER	OFF	ON	OFF	OFF
SOLAR	OFF	OFF	ON	OFF
SOLAR_PREFERRED	OFF	OFF	OFF	ON

processSpaHeatStatusEvent		
State	**Channel 189**	**Channel 190**
OFF	OFF	OFF
HEATER	ON	OFF
SOLAR	OFF	ON

processSpaJetsEvent				
State	**Channel 182**	**Channel 183**	**Channel 184**	**Channel 185**
OFF	ON	OFF	OFF	OFF
LOW	OFF	ON	OFF	OFF
MEDIUM	OFF	OFF	ON	OFF
HIGH	OFF	OFF	OFF	ON

Pre Amp Surround Sound Processor

- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- IPreAmpComponent
- IPreAmpComponentListener
- IPreAmpSurroundSoundProcessorComponent
- IPreAmpSurroundSoundProcessorComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Pre Amp Surround Sound Processor						
Interface: IPreAmpSurroundSoundProcessor						
Pre Amp Surround Sound Processor Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton (THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton (UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton (PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton (SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton (INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
adjustBalance(1)	IPreAmpComponent	164			BALANCE_UP	Ramping Channel: Balance is incremented when channel is activated
adjustBalance(-1)	IPreAmpComponent	165			BALANCE_DN	Ramping Channel: Balance is decremented when channel is activated
adjustBass(1)	IPreAmpComponent	166			BASS_UP	Ramping Channel: Bass is incremented when channel is activated
adjustBass(-1)	IPreAmpComponent	167			BASS_DN	Ramping Channel: Bass is decremented when channel is activated
adjustEqualizer(band,offset)	IPreAmpComponent			EQUALIZER_OFFSET-(band)=(offset)		Adjust Equalizer Band, given (band) number is a sequential index, and (offset) adjusts its gain, where (offset) ranges from -255 to 255
adjustEqualizer(band[],offset[])	IPreAmpComponent			EQUALIZER_OFFSET-(band)=(offset) [,(band)=(offset)]+		Adjust Equalizer Bands, given each (band) number, with its (offset) to adjust its gain, where (offset)s range from -255 to 255
adjustTreble(1)	IPreAmpComponent	168			TREBLE_UP	Ramping Channel: Treble is incremented when channel is activated
adjustTreble(-1)	IPreAmpComponent	169			TREBLE_DN	Ramping Channel: Treble is decremented when channel is activated
cycleLoudness()	IPreAmpComponent	206			LOUDNESS	Momentary Function Channel: Cycle loudness when channel is activated
getEqualizer()	IPreAmpComponent			?EQUALIZER		Query Equalizer for all supported (bands) and their (gains). Returns EQUALIZER-(band)=(gain) [,(band)=(gain)]+ where (band) ranges from 1 to 255 and its (gain) ranges from 0 to 255
getEqualizer(band)	IPreAmpComponent			?EQUALIZER-(band)		Query Equalizer for the (gain) of a given (band) number, where (band) number ranges from 1 to 255. Returns a format of EQUALIZER-(band)=(gain) where (gain) ranges from 0 to 255
getEqualizer(band[])	IPreAmpComponent			?EQUALIZER-(band) [,(band)]+		Query Equalizer for all given (bands). Returns EQUALIZER-(band)=(gain)[,(band)=(gain)]+ where (band) ranges from 1 to 255 and its corresponding (gain) ranges from 0 to 255

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getEqualizerBands()	IPreAmpComponent			?EQUALIZER_BANDS		Query Equalizer for all its supported (bands), where (band) numbers range from 1 to 255. Returns EQUALIZER_BANDS-(band)[,(band)]+
getSurroundMode()	IPreAmpComponent			?SURROUND		Query surround mode, responds with SURROUND-<mode>, where <mode> is MOVIE,MUSIC,OFF
nextSurroundMode()	IPreAmpComponent	170			SURROUND_NEXT	Momentary Function Channel: Next surround mode is selected when channel is activated
previousSurroundMode()	IPreAmpComponent	171			SURROUND_PREV	Momentary Function Channel: Previous surround mode is selected when channel is activated
setBalance(balance)	IPreAmpComponent		2		BALANCE_LVL	Set balance level, range is -128 to 128, -128 is left and 128 is right
setBass(bass)	IPreAmpComponent		3		BASS_LVL	Set bass level, range is 0-255
setEqualizer(band,gain)	IPreAmpComponent			EQUALIZER-(band)=(gain)		Set Equalizer bands gain, given a (band) number and (gain), where (band) ranges from1 to 255, and (gain) ranges from 0 to 255'
setEqualizer(band[],gain[])	IPreAmpComponent			EQUALIZER-(band)=(gain)[,(band)=(gain)]+		Set Equalizer, given each (band) number which ranges from 1 to 255, and its corresponding (gain), which ranges from 0 to 255
setLoudnessOn(state)	IPreAmpComponent	207			LOUDNESS_ON	Discrete Function Channel: Loudness is on while channel is active
setSurroundMode(sm)	IPreAmpComponent			SURROUND-<mode>		Set surround mode, where <mode> is MOVIE,MUSIC, OFF
setTreble(treble)	IPreAmpComponent		4		TREBLE_LVL	Set treble level, range is 0-255
processBalanceEvent(int)	IPreAmpComponentListener		2		BALANCE_LVL	Balance changed, range is -128 to 128, -128 is left and 128 is right
processBassEvent(int)	IPreAmpComponentListener		3		BASS_LVL	Bass changed, range is 0-255
processEqualizerEvent(int[][])	IPreAmpComponentListener			EQUALIZER-(band)=(gain)[,(band)=(gain)]+		The (gain) changed for the Equalizer at a given (band) number, where (band) ranges from 1 to 255 and its (gain) ranges from 0 to 255
processLoudnessEvent(boolean)	IPreAmpComponentListener	207			LOUDNESS_FB	Feedback Channel: Loudness is on if channel is on
processSurroundModeEvent (SurroundMode)	IPreAmpComponentListener			SURROUND-<mode>		Surround mode changed, where <mode> is MUSIC,MOVIE,OFF
processTrebleEvent(int)	IPreAmpComponentListener		4		TREBLE_LVL	Treble changed, range is 0-255
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputPropertiesEvent (String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> [;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputPropertyEvent (String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)

Pre Amp Surround Sound Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent (boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent (UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent (DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Pre Amp Surround Sound Processor State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Receiver

- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- IPreAmpComponent
- IPreAmpComponentListener
- IReceiverComponent
- IReceiverComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- ITunerStationComponent
- ITunerStationComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Receiver						
Interface: IReceiver						
Receiver Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>, <channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
adjustBalance(1)	IPreAmpComponent	164			BALANCE_UP	Ramping Channel: Balance is incremented when channel is activated
adjustBalance(-1)	IPreAmpComponent	165			BALANCE_DN	Ramping Channel: Balance is decremented when channel is activated
adjustBass(1)	IPreAmpComponent	166			BASS_UP	Ramping Channel: Bass is incremented when channel is activated
adjustBass(-1)	IPreAmpComponent	167			BASS_DN	Ramping Channel: Bass is decremented when channel is activated
adjustEqualizer(band,offset)	IPreAmpComponent			EQUALIZER_OFFSET-(band)=(offset)		Adjust Equalizer Band, given (band) number is a sequential index, and (offset) adjusts its gain, where (offset) ranges from -255 to 255
adjustEqualizer(band[],offset[])	IPreAmpComponent			EQUALIZER_OFFSET-(band)=(offset) [,(band)=(offset)]+		Adjust Equalizer Bands, given each (band) number, with its (offset) to adjust its gain, where (offset)s range from -255 to 255
adjustTreble(1)	IPreAmpComponent	168			TREBLE_UP	Ramping Channel: Treble is incremented when channel is activated
adjustTreble(-1)	IPreAmpComponent	169			TREBLE_DN	Ramping Channel: Treble is decremented when channel is activated
cycleLoudness()	IPreAmpComponent	206			LOUDNESS	Momentary Function Channel: Cycle loudness when channel is activated
getEqualizer()	IPreAmpComponent			?EQUALIZER		Query Equalizer for all supported (bands) and their (gains). Returns EQUALIZER-(band)=(gain)[,(band)=(gain)]+ where (band) ranges from 1 to 255 and its (gain) ranges from 0 to 255
getEqualizer(band)	IPreAmpComponent			?EQUALIZER-(band)		Query Equalizer for the (gain) of a given (band) number, where (band) number ranges from 1 to 255. Returns a format of EQUALIZER-(band)=(gain) where (gain) ranges from 0 to 255
getEqualizer(band[])	IPreAmpComponent			?EQUALIZER-(band)[,(band)]+		Query Equalizer for all given (bands). Returns EQUALIZER-(band)=(gain) [,(band)=(gain)]+ where (band) ranges from 1 to 255 and its corresponding (gain) ranges from 0 to 255
getEqualizerBands()	IPreAmpComponent			?EQUALIZER_BANDS		Query Equalizer for all its supported (bands), where (band) numbers range from 1 to 255. Returns EQUALIZER_BANDS-(band)[,(band)]+

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getSurroundMode()	IPreAmpComponent			?SURROUND		Query surround mode, responds with SURROUND-<mode>, where <mode> is MOVIE,MUSIC,OFF
nextSurroundMode()	IPreAmpComponent	170			SURROUND_NEXT	Momentary Function Channel: Next surround mode is selected when channel is activated
previousSurroundMode()	IPreAmpComponent	171			SURROUND_PREV	Momentary Function Channel: Previous surround mode is selected when channel is activated
setBalance(balance)	IPreAmpComponent		2		BALANCE_LVL	Set balance level, range is -128 to 128, -128 is left and 128 is right
setBass(bass)	IPreAmpComponent		3		BASS_LVL	Set bass level, range is 0-255
setEqualizer(band,gain)	IPreAmpComponent			EQUALIZER-(band)=(gain)		Set Equalizer bands gain, given a (band) number and (gain), where (band) ranges from 1t o 255, and (gain) ranges from 0 to 255'
setEqualizer(band[],gain[])	IPreAmpComponent			EQUALIZER-(band)=(gain) [,(band)=(gain)]+		Set Equalizer, given each (band) number which ranges from 1 to 255, and its corresponding (gain), which ranges from 0 to 255
setLoudnessOn(state)	IPreAmpComponent	207			LOUDNESS_ON	Discrete Function Channel: Loudness is on while channel is active
setSurroundMode(sm)	IPreAmpComponent			SURROUND-<mode>		Set surround mode, where <mode> is MOVIE,MUSIC,OFF
setTreble(treble)	IPreAmpComponent		4		TREBLE_LVL	Set treble level, range is 0-255
processBalanceEvent(int)	IPreAmpComponentListener		2		BALANCE_LVL	Balance changed, range is -128 to 128, -128 is left and 128 is right
processBassEvent(int)	IPreAmpComponentListener		3		BASS_LVL	Bass changed, range is 0-255
processEqualizerEvent(int[][])	IPreAmpComponentListener			EQUALIZER-(band)=(gain) [,(band)=(gain)]+		The (gain) changed for the Equalizer at a given (band) number, where (band) ranges from 1 to 255 and its (gain) ranges from 0 to 255
processLoudnessEvent(boolean)	IPreAmpComponentListener	207			LOUDNESS_FB	Feedback Channel: Loudness is on if channel is on
processSurroundModeEvent (SurroundMode)	IPreAmpComponentListener			SURROUND-<mode>		Surround mode changed, where <mode> is MUSIC,MOVIE,OFF
processTrebleEvent(int)	IPreAmpComponentListener		4		TREBLE_LVL	Treble changed, range is 0-255

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signal-Type>,<deviceLabel>,<display-Name>[;<index>,<inputGroup>,<signalTyp e>,<deviceLabel>,<displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputPropertiesEvent(String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>[; <index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>]*		Returns comma-separated string contain-ing <index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processInputPropertyEvent(String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cycleBand()	ITunerStationComponent	40			TUNER_BAND	Momentary Function Channel: Cycle tuner band when channel is activated
cycleDisplayInfo()	ITunerStationComponent	234			TUNER_OSD	Momentary Function Channel: Cycle on-screen or front-panel display info when channel is activated
cycleStationPresetGroup()	ITunerStationComponent	224			TUNER_PRESET_GROUP	Momentary Function Channel: Cycle station preset group/bank when channel is activated
decrementStation()	ITunerStationComponent	226			TUNER_STATION_DN	Momentary Function Channel: Station is decremented when channel is activated
getBand()	ITunerStationComponent			?BAND		Query for band, responds with BAND-<tb> where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV
getStation()	ITunerStationComponent			?XCH		Query for station, responds with XCH-<station> where <station> is a station string such as "501", "103.7" or "5.1"
getStationPreset()	ITunerStationComponent			?TUNERPRESET		Query for tuner preset, responds with TUNERPRESET-<preset>
getStationPresetCount()	ITunerStationComponent			?STATIONPRESET COUNT		Query for number of valid station presets, responds with STATIONPRESETCOUNT-<count>
getStationPresetProperties()	ITunerStationComponent			?STATIONPRESET PROPERTIES		Query input properties for all station presets, responds with STATIONPRESETPROPERTIES-<index>,<displayName>,<station> [;<index>,<displayName>,<station>]*
getStationPresetProperty(index)	ITunerStationComponent			?STATIONPRESET PROPERTY-<index>		Query properties for a single station preset, responds with STATIONPRESETPROPERTY-<index>,<displayName>,<station>

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getStationPresetSelect()	ITunerStationComponent			?STATIONPRESET SELECT		Query for the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
getTunerBandProperties()	ITunerStationComponent			?TUNERBAND PROPERTIES		Query input properties for all tuner bands, responds with TUNERBANDPROPERTIES-<index>, <displayName>,<tb>[,<index>, <displayName>,<tb>]*
getTunerBandProperty(index)	ITunerStationComponent			?TUNERBANDPROPERTY-<index>		Query properties for a single tuner band, responds with TUNERBANDPROPERTY-<index>,<displayName>,<tb>
getTunerBandPropertyCount()	ITunerStationComponent			?TUNERBANDPROPERTY COUNT		Query for number of valid tuner bands, responds with TUNERBANDPROPERTYCOUNT-<count>
getTunerBandSelect()	ITunerStationComponent			?TUNERBANDSELECT		Query for the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
getTunerComponentProperties()	ITunerStationComponent			?TUNERCOMPONENT PROPERTIES		Query properties for a single tuner component, responds with TUNERCOMPONENTPROPERTIES-<index>,<displayName>,<value> [;<index>,<displayName>,<value>]*
getTunerComponentProperty()	ITunerStationComponent			?TUNERCOMPONENT PROPERTY		Query input properties for all inputs, responds with TUNERCOMPONENTPROPERTY-<index>,<displayName>,<value>
getTunerComponentPropertyCount()	ITunerStationComponent			?TUNERCOMPONENT PROPERTYCOUNT		Query for number of tuner components, responds with TUNERCOMPONENTPROPERTY-COUNT-<count>
getTunerComponentSelect()	ITunerStationComponent			?TUNERCOMPONENT SELECT		Query for the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
gotoPreviousStation()	ITunerStationComponent	235			TUNER_PREV	Momentary Function Channel: Previous station is selected when channel is activated
incrementStation()	ITunerStationComponent	225			TUNER_STATION_UP	Momentary Function Channel: Station is incremented when channel is activated
nextStationPreset()	ITunerStationComponent	22			CHAN_UP	Momentary Function Channel: Next station preset is selected when channel is activated

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
previousStationPreset()	ITunerStationComponent	23			CHAN_DN	Momentary Function Channel: Previous station preset is selected when channel is activated
scanStation(FORWARD)	ITunerStationComponent	227			TUNER_SCAN_FWD	Momentary Function Channel: Scans for next station while channel is activate.
scanStation(BACKWARD)	ITunerStationComponent	228			TUNER_SCAN_REV	Momentary Function Channel: Scans for previous station while channel is activate
seekStation(FORWARD)	ITunerStationComponent	229			TUNER_SEEK_FWD	Momentary Function Channel: Seeks for next station while channel is activate
seekStation(BACKWARD)	ITunerStationComponent	230			TUNER_SEEK_REV	Momentary Function Channel: Seeks for previous station while channel is activate
setBand(tb)	ITunerStationComponent			BAND-<tb>		Set band, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO,LONG_WAVE,MEDIUM_WAVE,SHORT_WAVE,TV
setStation(station)	ITunerStationComponent			XCH-<station>		Set station, where <station> is a station string such as "501", "103.7" or "5.1"
setStationPreset(preset)	ITunerStationComponent			TUNERPRESET-<preset>		Recall tuner preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setStationPresetSelect(index)	ITunerStationComponent			STATIONPRESETSELECT-<index>		Set the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
setTunerBandSelect(index)	ITunerStationComponent			TUNERBANDSELECT-<index>		Set the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
setTunerComponentSelect(index)	ITunerStationComponent			TUNERCOMPONENT SELECT-<index>		Set the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
processBandEvent(TunerBand)	ITunerStationComponentListener			BAND-<band>		Band changed, where <tb> is AM,FM, FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV
processStationEvent(String)	ITunerStationComponentListener			XCH-<station>		Station changed, where <station> is a station string such as "501", "103.7" or "5.1"
processStationPresetCountEvent(count)	ITunerStationComponentListener			STATIONPRESETCOUNT-<count>		Response to ?TUNERCOUNT, where <count> is the integer number of tuner components
processStationPresetEvent(int)	ITunerStationComponentListener			TUNERPRESET-<int>		Tuner preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processStationPresetPropertiesEvent (String[])	ITunerStationComponentListener			STATIONPRESET PROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>,<value>;...]		Returns the properties for each station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetPropertyEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTY-<index>,<displayName>, <value>		Returns the properties for a single station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetSelectEvent(index)	ITunerStationComponentListener			STATIONPRESETSELECT-<index>		Returns the selected station preset index
processTunerBandPropertiesEvent(String[])	ITunerStationComponentListener			TUNERBANDPROPERTIES-<index>,<displayName>, <tunerBand>[;<index>, <displayName>, <tunerBand>;...]		Returns properties for each tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandPropertyCountEvent (count)	ITunerStationComponentListener			TUNERBANDPROPERTY COUNT-<count>		Returns the number of valid tuner bands
processTunerBandPropertyEvent(String[])	ITunerStationComponentListener			TUNERBANDPROPERTY-<index>,<displayName>, <tunerBand>		Returns properties for a single tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandSelectEvent(index)	ITunerStationComponentListener			TUNERBANDSELECT-<index>		Returns the selected tuner band index
processTunerComponentPropertiesEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>,<value>;...]		Returns the properties for each tuner component
processTunerComponentPropertyCountEvent (count)	ITunerStationComponentListener			TUNERCOMPONENT PROPERTYCOUNT		Returns the integer number of tuner components
processTunerComponentPropertyEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single tuner component
processTunerComponentSelectEvent(index)	ITunerStationComponentListener			TUNERCOMPONENT SELECT-<index>		Returns the selected tuner component index
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated

Receiver Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Receiver State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Relay Device

- IModuleComponent
- IModuleComponentListener
- IRelayDeviceComponent
- IRelayDeviceComponentListener

Name: Relay Device						
Interface: IRelayDevice						
Relay Device Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on

Relay Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
getRelayChannelCount()	IRelayDeviceComponent			?RELAYCHANNELCOUNT		Query for the number of channels on a relay device, responds with RELAYCHANNELCOUNT
getRelayChannelState(rly-chan)	IRelayDeviceComponent			?RELAYCHANNELSTATE-<channel>		Query for the relay channel state, where <rly-chan> is the integer channel number, responds with RELAYCHANNELCOUNT
setRelayChannelState (rly-chan,rly-state)	IRelayDeviceComponent			RELAYCHANNELSTATE-<channel>,<state>		Sets the relay channel state, where <rly-chan> is the integer channel number and <rly-state> is ON or OFF
processRelayChannelCountEvent (count)	IRelayDeviceComponentListener			RELAYCHANNELCOUNT-<count>		Response to ?RELAYCHANNELCOUNT, where <count> is an integer value
processRelayChannelStateEvent (rly-state)	IRelayDeviceComponentListener			RELAYCHANNELSTATE-<channel>,<state>		Response to ?RELAYCHANNELSTATE, where <rly-chan> is the integer channel number and <rly-state> is ON or OFF

RFIDSystem

Name: RFIDSystem						
Interface: IRFIDSystem						
RFID System Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent (byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

RFID System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getReaders()	IRFIDSystemComponent			?READERS		Request a list of all the RF Readers in the configured Reader Network controlled by the Duet Module. Responds with READERS-<readerAddress1>[,<readerAddressN>]* where <readerAddress> is a user configured String associated with a specific RF Reader in an RF Reader Network. The user configures the <readerAddress> and uses it in their NetLinx logic. The Duet Module maps the <readerAddress> to the RF Reader's device specific address. <readerAddress> is a String, the default unconfigured value is the RF Reader's device specific address.
getReadersByTag(tagId)	IRFIDSystemComponent			?READERSBYTAG-<tagId>	getReadersByTag(tagId)	Request the RF Readers that currently have the given <tagId> acquired. Multiple responses may be sent to support a Tag that is acquired by many RF Readers. The <readerCount> always shows the total number of RF Reader sets in the response, where an RF Reader set consists of the <readerAddress> and <tagSignalStrength>. Responds with READERSBYTAG<tagId>,<tagName>,<tagInfo>,<tagPercentPower>,<readerCount>,<readerIndex>[,<readerAddress>,<tagSignalStrength>]* where <tagId> is a unique identifier assigned to the Tag by the RF Tag manufacturer, <tagId> is a String; <tagName> is a user configured friendly name associated with the <tagId>. The Duet Module maps <tagId> to the <tagName>. <tagName> is a String, truncated to a 20 character limit. The default unconfigured value is <tagId>; <tagInfo> is a user configured String further defining the associated <tagId>. The Duet Module maps the <tagId> to the <tagInfo>. The <tagInfo> can be used to refine type-based processing. For example the user can configure <tagInfo> to describe assets such as PortableAsset, FixedAsset, Employee, Serviceperson, or Guest. <tagInfo> could also describe the type of Tag as Badge, Keychain, or Tag. <tagInfo> is a String, truncated to a 20 character limit. The default unconfigured value is its <tagId>; <tagPercentPower> is the percent remaining battery power for the <tagId> acquired. <tagPercentPower> is an Integer from 1% to 100%; <readerCount> is the total number of RF Reader sets in the READERSBYTAG response. Possible values are: 0 = no RF Readers currently see this Tag; or positive number N, where there will be N number of RF Reader sets in the response.

RFID System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
						If the value is 0, no RF Reader sets follow. <readerCount> is an Integer from 0 to the maximum number of RF Readers supported by the Duet Module; <readerIndex> is used to support multiple responses to the ?READERSBYTAG command. <readerIndex> gives the index of the first RF Readerset in the current message. For example, if <readerCount> is 100 the message may be broken up into two responses, each with 50 RF Reader sets. The first response will have <readerCount> = 100 and <readerIndex> = 1. The second response will maintain <readerCount> = 100 and set <readerIndex> = 51. RF Reader sets consist of 2 comma-separated-values: <readerAddress>,<tagSignalStrength>. <readerIndex> is an Integer from 1 to <readerCount>; <readerAddress> is a user configured String associated with a specific RF Reader in an RF Reader Network. The user configures the <readerAddress> and uses it in their NetLinx logic. The Duet Module maps the <readerAddress> to the RF Reader's device specific address. <readerAddress> is a String, the default unconfigured value is the RF Reader's device specific address; and <tagSignalStrength> is the Signal Strength of the associated <tagId> recorded by the RF Reader associated with <readerAddress>. <tagSignalStrength> is an Integer from 0 to 255. <tagSignalStrength> is synonymous with the term RSSI or RF SignalStrength Indicator.
setReaderAutoPoll (readerAddress, autoPoll, interval)	IRFIDSystemComponent			READERAUTOPOLL-<readerAddress>, <autoPoll>[,<interval>]		<autoPoll> is a feature of an RF Reader at <readerAddress> that can be enabled (ON) or disabled (OFF). When the AutoPoll TSE is ON, then ?TAGSBY-READER is sent, per the optional time <interval> where <readerAddress> is a user configured String associated with a specific RF Reader in an RF Reader Network. The user configures the <readerAddress> and uses it in their NetLinx logic. The Duet Module maps the <readerAddress> to the RF Reader's device specific address. <readerAddress> is a String, the default unconfigured value is the RF Reader's device specific address.

RFID System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processReadersByTagEvent (String,String,String, int,int,int,String,int)	IRFIDSystemComponentListener			READERSBYTAG-<tagId>,<tagName>,<tagInfo>, <tagPercentPower>, <readerCount>, <readerIndex> [,<readerAddress>, <tagSignalStrength>]*		Sends READERSBYTAG-<tagId>,<tagName>, <tagInfo>,<tagPercentPower>,<readerCount>, <readerIndex> [,<readerAddress>, <tagSignalStrength>]* where each <readerAddress> currently acquiring <tagId> is represented as a 2 item Reader set: <readerAddress>, <tagSignalStrength>. The Duet Module is responsible for maintaining the RF Readers currently acquiring each Tag, and for mapping the RF Reader device address to the user configured <readerAddress>.
processReadersEvent(String)	IRFIDSystemComponentListener			READERS-<readerAddress1> [,readerAddressN>]*		Sends READERS-<readerAddress1>[,<readerAddressN>]* as the list of N <readerAddress>s in the configured RF Reader Network controlled by the Duet Module. The Duet Module is responsible for knowing the RF Readers participating in the RF Reader Network under its control, and for mapping the RF Reader device address to the user configured <readerAddress>.
processReaderStatusEvent (String,ReaderStatus,int)	IRFIDSystemComponentListener			READERSTATUS-<readerAddress>, <readerStatus>, <errorCount>		Sends READERSTATUS-<readerAddress>, <readerStatus>,<errorCount> for the RF Reader associated with the given <readerAddress>.
processTagAcquiredEvent (String,String,String, String,String,int,int)	IRFIDSystemComponentListener			TAGACQUIRED-<readerAddress>,<tagId>, <tagName>,<tagInfo>, <timestamp>, <tagSignalStrength>, <tagPercentPower>		Unsolicited notification event that the RF Reader associated with <readerAddress> has just acquired the signal of the RF Tag associated with <tagId>. The Duet Module is responsible for mapping the RF Reader device address to the user configured <readerAddress>. The Duet Module may choose to add configuration to enable the user to filter out this event. <timestamp> is when the RF Tag signal was acquired. <timestamp> is a system time, formatted typically as a numeric String yyyymmddhhmissnnn (nnn is up to msec system granularity).
processTagButtonEvent (String,String,String,TagButton)	IRFIDSystemComponentListener			TAGBUTTON-<tagId>,<tagName>, <tagInfo>,<state>		Unsolicited notification event that the button on the Tag associated with <tagId> was pushed or released. If a single button push or release event is received by more than one RF Reader, the Duet Module is responsible for consolidating the report from all RF Readers into a single TAGBUTTON event given to the NetLinx program. <state> is a TSE with values of PUSH or RELEASE.
processTagInfoEvent (String,String,String)	IRFIDSystemComponentListener			TAGINFO-<tagId>, <tagName>,<tagInfo>		Sends TAGINFO-<tagId>,<tagName>,<tagInfo> for the RF Tag associated with the given <tagId>.

RFID System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processTagLostEvent (String,String,String, String,String)	IRFIDSystemComponentListener			TAGLOST-<readerAddress>, <tagId>, <tagName>, <tagInfo>,<lastTimestamp>		Unsolicited notification event that the RF Reader associated with <readerAddress> has just lost the signal of the RF Tag associated with <tagId>. The Duet Module is responsible for mapping the RF Reader device address to the user configured <readerAddress>. The Duet Module may choose to add configuration to enable the user to filter out this event. <lastTimestamp> is when the last valid <tagSignalStrength> was acquired for the RF Tag associated with <tagId> by this Reader. <lastTimestamp> is a system timestamp.
processTagsByReaderEvent (String,int,int,String,String, String,int,int)	IRFIDSystemComponentListener			TAGSBYREADER-<readerAddress>,<tagCount>, <tagIndex> [,<tagId>, <tagName>,<tagInfo>, <tagSignalStrength>, <tagPercentPower>]*		Sends TAGSBYREADER-<readerAddress>,<tagCount>,<tagIndex>[,<tagId>,<tagName>,<tagInfo>,<tagSignalStrength>, <tagPercentPower>]* where each <tagId>, currently acquired by the RF Reader associated with <readerAddress>, is represented as a 5 item Tag set: <tagId>,<tagName>,<tagInfo>,<tagSignalStrength>,<tagPercentPower>. The Duet Module is responsible for maintaining the Tags currently acquired by each RF Reader, and for mapping the RF device address to the user configured <readerAddress>.
processTagSignalStrengthEvent (String,String,String,String, String,int,int)	IRFIDSystemComponentListener			TAGSIGNALSTRENGTH-<readerAddress>,<tagId>, <tagName>,<tagInfo>, <timestamp>, <tagSignalStrength>, <tagPercentPower>		Unsolicited notification event that the RF Reader associated with <readerAddress> received a <tagSignalStrength> for the RF Tag associated with <tagId> that is different from its last known <tagSignalStrength>. The Duet Module is responsible for maintaining the last signal strength reported by each RF Reader for each Tag it has acquired. The Duet Module can then check the stored signal strength against the latest reported signal strength to see if it has changed. The Duet Module may choose to add configuration to enable the user to filter out this event.

Security System

- IModuleComponent
- IModuleComponentListener
- ISecuritySystemComponent
- ISecuritySystemComponentListener

Name: Security System						
Interface: ISecuritySystem						
Security System Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy><hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When pass-back is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

Security System Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getPointState(secPoint)	ISecuritySystemComponent			?SECPOINTSTATE-<point>		Query for Point State, responds with SECPOINTSTATE-<point>,<state> where <point> is 1 to the maximum point point as returned by getPointCount() (see specific module documentation) and <state> is BYPASS or ACTIVE
getPointStatus(secPoint)	ISecuritySystemComponent			?SECPOINTSTATUS-<point>		Query for Point Status, responds with SECPOINTSTATUS-<point>,<status> where <point> is 1 to the maximum point point as returned by getPointCount() (see specific module documentation) and <status> is ACTIVE, FAULT or BYPASSED
getSecurityState()	ISecuritySystemComponent			?SECSTATE		Query for Security State, responds with SECSTATE-<state> where <state> is ARM_HOME, ARM, ARM_HOME_NOW, ARM_NOW, DISARM, FIRE, PANIC, POLICE, MEDICAL or NONE
getSecurityStatus()	ISecuritySystemComponent			?SECSTATUS		Query for Security Status, responds with SECSTATUS-<status> where <status> is DISARMED, ARMED_HOME, ARMED, ALARM
isOKToArm()	ISecuritySystemComponent			?SECARMABLE		Query for Security arm-able status, responds with SECARMABLE-<status> where <status> is 1 or 0
setPointState(secPoint,ps,pw)	ISecuritySystemComponent			SECPOINTSTATE-<point>,<state>, <password>		Set Point State where <point> is 1 to the maximum point as returned by getPointCount() (see specific module documentation), <state> is ACTIVE or BYPASS and <password> is the password required to complete the operation (see specific module documentation).
setSecurityState(ss,password)	ISecuritySystemComponent			SECSTATE-<state>, <password>		Set Security State where <state> is ARM_HOME, ARM, ARM_HOME_NOW, ARM_NOW, DISARM, FIRE, PANIC, POLICE, MEDICAL and <password> is the password required to complete the operation (see specific module documentation).
processOKToArmEvent (boolean)	ISecuritySystemComponentListener			SECARMABLE-<status>		Security arm-able changed, where <status> is 1 or 0, 1 means the security system can be armed.
processPointStateEvent (SecurityPointState)	ISecuritySystemComponentListener			SECPOINTSTATE-<point>,<state>		Point State changed, where <point> is 1 to the maximum point as returned by getPointCount() (see specific module documentation) and <state> is BYPASS or ACTIVE
processPointStatusEvent (SecurityPointStatus)	ISecuritySystemComponentListener			SECPOINTSTATUS-<point>,<status>		Point Status changed, where <point> is 1 to the maximum point as returned by getPointCount() (see specific module documentation) and <status> is ACTIVE, FAULT or BYPASSED
processSecurityStateEvent (SecurityState)	ISecuritySystemComponentListener			SECSTATE-<state>		Security State changed, where <state> is ARM_HOME, ARM, ARM_HOME_NOW, ARM_NOW, DISARM, FIRE, PANIC, POLICE, MEDICAL or NONE
processSecurityStatusEvent (SecurityStatus)	ISecuritySystemComponentListener			SECSTATUS-<status>		Security Status changed, where <status> is DISARMED, ARMED_HOME, ARMED, ALARM

Sensor Device

- IModuleComponent
- IModuleComponentListener
- ISensorComponent
- ISensorComponentListener

Name: Sensor Device						
Interface: ISensorDevice						
Sensor Device Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent (byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
processSensorStateEvent (ON)	ISensorComponentListener	255			SENSOR_FB	Feedback Channel: Sensor state changed, sensor is on while channel is on

Sensor Device Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processSensorValueEvent (int)	ISensorComponentListener		7		SENSOR_VALUE	Value of the sensor changed, range is specific to the sensor type (see specific module documentation)

Sensor Device State Charts

processSensorStateEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Settop Box

- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ISettopBoxComponent
- ISettopBoxComponentListener
- ITunerStationComponent
- ITunerStationComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Settop Box

Interface: ISettopBox

Settop Box Functions

Name	Component	Channel	Level	Command	Constant	Notes
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy><hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>, <channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleABSwitch()	ISettopBoxComponent	42			CABLE_AB	Momentary Function Channel: Cycle AB switch when channel is activated
setBSwitchOn(state)	ISettopBoxComponent	212			CABLE_B_ON	Discrete Function Channel: AB switch set to B when channel is active
processBSwitchEvent(boolean)	ISettopBoxComponentListener	212			CABLE_B_FB	Feedback Channel: AB switch set to B when channel is on
cycleBand()	ITunerStationComponent	40			TUNER_BAND	Momentary Function Channel: Cycle tuner band when channel is activated
cycleDisplayInfo()	ITunerStationComponent	234			TUNER_OSD	Momentary Function Channel: Cycle on-screen or front-panel display info when channel is activated
cycleStationPresetGroup()	ITunerStationComponent	224			TUNER_PRESET_GROUP	Momentary Function Channel: Cycle station preset group/bank when channel is activated
decrementStation()	ITunerStationComponent	226			TUNER_STATION_DN	Momentary Function Channel: Station is decremented when channel is activated
getBand()	ITunerStationComponent			?BAND		Query for band, responds with BAND-<tb> where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getStation()	ITunerStationComponent			?XCH		Query for station, responds with XCH-<station> where <station> is a station string such as "501", "103.7" or "5.1"
getStationPreset()	ITunerStationComponent			?TUNERPRESET		Query for tuner preset, responds with TUNERPRESET-<preset>
getStationPresetCount()	ITunerStationComponent			?STATIONPRESET COUNT		Query for number of valid station presets, responds with STATIONPRESETCOUNT-<count>
getStationPresetProperties()	ITunerStationComponent			?STATIONPRESET PROPERTIES		Query input properties for all station presets, responds with STATIONPRESETPROPERTIES-<index>,<displayName>,<station> [;<index>,<displayName>,<station>]*
getStationPresetProperty(index)	ITunerStationComponent			?STATIONPRESET PROPERTY-<index>		Query properties for a single station preset, responds with STATIONPRESETPROPERTY-<index>,<displayName>,<station>
getStationPresetSelect()	ITunerStationComponent			?STATIONPRESET SELECT		Query for the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
getTunerBandProperties()	ITunerStationComponent			?TUNERBAND PROPERTIES		Query input properties for all tuner bands, responds with TUNERBANDPROPERTIES-<index>,<displayName>,<tb>[,<index>, <displayName>,<tb>]*
getTunerBandProperty(index)	ITunerStationComponent			?TUNERBAND PROPERTY-<index>		Query properties for a single tuner band, responds with TUNERBANDPROPERTY-<index>,<displayName>,<tb>
getTunerBandPropertyCount()	ITunerStationComponent			?TUNERBAND PROPERTYCOUNT		Query for number of valid tuner bands, responds with TUNERBANDPROPERTYCOUNT-<count>
getTunerBandSelect()	ITunerStationComponent			?TUNERBAND SELECT		Query for the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
getTunerComponentProperties()	ITunerStationComponent			?TUNERCOMPONENT PROPERTIES		Query properties for a single tuner component, responds with TUNERCOMPONENTPROPERTIES-<index>,<displayName>,<value>[;<index>, <displayName>,<value>]*
getTunerComponentProperty()	ITunerStationComponent			?TUNERCOMPONENT PROPERTY		Query input properties for all inputs, responds with TUNERCOMPONENTPROPERTY-<index>,<displayName>,<value>
getTunerComponentPropertyCount()	ITunerStationComponent			?TUNERCOMPONENT PROPERTYCOUNT		Query for number of tuner components, responds with TUNERCOMPONENTPROPERTYCOUNT-<count>
getTunerComponentSelect()	ITunerStationComponent			?TUNERCOMPONENT SELECT		Query for the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
gotoPreviousStation()	ITunerStationComponent	235			TUNER_PREV	Momentary Function Channel: Previous station is selected when channel is activated
incrementStation()	ITunerStationComponent	225			TUNER_STATION_UP	Momentary Function Channel: Station is incremented when channel is activated
nextStationPreset()	ITunerStationComponent	22			CHAN_UP	Momentary Function Channel: Next station preset is selected when channel is activated
previousStationPreset()	ITunerStationComponent	23			CHAN_DN	Momentary Function Channel: Previous station preset is selected when channel is activated
scanStation(FORWARD)	ITunerStationComponent	227			TUNER_SCAN_FWD	Momentary Function Channel: Scans for next station while channel is activate.
scanStation(BACKWARD)	ITunerStationComponent	228			TUNER_SCAN_REV	Momentary Function Channel: Scans for previous station while channel is activate
seekStation(FORWARD)	ITunerStationComponent	229			TUNER_SEEK_FWD	Momentary Function Channel: Seeks for next station while channel is activate
seekStation(BACKWARD)	ITunerStationComponent	230			TUNER_SEEK_REV	Momentary Function Channel: Seeks for previous station while channel is activate
setBand(tb)	ITunerStationComponent			BAND-<tb>		Set band, where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV
setStation(station)	ITunerStationComponent			XCH-<station>		Set station, where <station> is a station string such as "501", "103.7" or "5.1"
setStationPreset(preset)	ITunerStationComponent			TUNERPRESET-<preset>		Recall tuner preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setStationPresetSelect(index)	ITunerStationComponent			STATIONPRESET SELECT-<index>		Set the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
setTunerBandSelect(index)	ITunerStationComponent			TUNERBANDSELECT-<index>		Set the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
setTunerComponentSelect(index)	ITunerStationComponent			TUNERCOMPONENT SELECT-<index>		Set the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
processBandEvent(TunerBand)	ITunerStationComponentListener			BAND-<band>		Band changed, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE,SHORT_WAVE, TV
processStationEvent(String)	ITunerStationComponentListener			XCH-<station>		Station changed, where <station> is a station string such as "501", "103.7" or "5.1"
processStationPresetCountEvent(count)	ITunerStationComponentListener			STATIONPRESET COUNT-<count>		Response to ?TUNERCOUNT, where <count> is the integer number of tuner components

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processStationPresetEvent(int)	ITunerStationComponentListener			TUNERPRESET-<int>		Tuner preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processStationPresetPropertiesEvent (String[])	ITunerStationComponentListener			STATIONPRESET PROPERTIES-<index>, <displayName>, <value>[;<index>, <displayName>, <value>;...]		Returns the properties for each station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetPropertyEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTY-<index>, <displayName>, <value>		Returns the properties for a single station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetSelectEvent(index)	ITunerStationComponentListener			STATIONPRESET SELECT-<index>		Returns the selected station preset index
processTunerBandPropertiesEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTIES-<index>, <displayName>, <tunerBand>[;<index>, <displayName>, <tunerBand>;...]		Returns properties for each tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandPropertyCountEvent (count)	ITunerStationComponentListener			TUNERBAND PROPERTYCOUNT-<count>		Returns the number of valid tuner bands
processTunerBandPropertyEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTY-<index>, <displayName>, <tunerBand>		Returns properties for a single tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandSelectEvent(index)	ITunerStationComponentListener			TUNERBANDSELECT-<index>		Returns the selected tuner band index
processTunerComponentPropertiesEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTIES-<index>, <displayName>, <value>[;<index>, <displayName>, <value>;...]		Returns the properties for each tuner component
processTunerComponentPropertyCountEvent (count)	ITunerStationComponentListener			TUNERCOMPONENT PROPERTYCOUNT		Returns the integer number of tuner components
processTunerComponentPropertyEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTY-<index>, <displayName>, <value>		Returns the properties for a single tuner component

Settop Box Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processTunerComponentSelectEvent(index)	ITunerStationComponentListener			TUNERCOMPONENT SELECT-<index>		Returns the selected tuner component index
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Settop Box State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Slide Projector

- ILampComponent
- ILampComponentListener
- IModuleComponent
- IModuleComponentListener
- ISlideProjectorComponent
- ISlideProjectorComponentListener

Name: Slide Projector						
Interface: ISlideProjector						
Slide Projector Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleLampPower()	ILampComponent	9			POWER	Momentary Function Channel: Cycle lamp power when channel is activated
getCoolDownTime()	ILampComponent			?COOLDOWN		Query for cool down time, responds with COOLDOWN-<time> where <time> is in seconds
getLampTime()	ILampComponent			?LAMPTIME		Query for lamp time, responds with LAMPTIME-<time> where <time> is in hours
getWarmUpTime()	ILampComponent			?WARMUP		Query for warm up time, responds with WARMUP-<time> where <time> is in seconds
setCoolDownTime(secs)	ILampComponent			COOLDOWN-<seconds>		Set cool down time where <time> is in seconds
setCounterNotificationOn(state)	ILampComponent			COUNTERNOTIFY-<state>		Turn counter notification on or off, where <state> is 1 or 0
setLampPower(ON)	ILampComponent	27			PWR_ON	Momentary Function Channel: Lamp power is turned on when channel is activated
setLampPower(OFF)	ILampComponent	28			PWR_OFF	Momentary Function Channel: Lamp power is turned off when channel is activated
setLampPower(ps)	ILampComponent	255			POWER_ON	Discrete Function Channel: Lamp power is on while channel is active
setLampTime(hours)	ILampComponent			LAMPTIME-<hours>		Set lamp time where <time> is in hours
setWarmUpTime(secs)	ILampComponent			WARMUP-<seconds>		Set warm up time where <time> is in seconds
processCoolDownCounterEvent (int)	ILampComponentListener			COOLING-<time>		Cool down counter time, <time> is seconds remaining
processLampPowerEvent (WARMING_UP)	ILampComponentListener	253			LAMP_WARMING_FB	Feedback Channel: Indicates Lamp is warming and cannot accept commands
processLampPowerEvent (COOLING_DOWN)	ILampComponentListener	254			LAMP_COOLING_FB	Feedback Channel: Indicates Lamp is cooling and cannot accept commands
processLampPowerEvent(ON)	ILampComponentListener	255			LAMP_POWER_FB	Feedback Channel: Indicates lamp is ON

Slide Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processLampTimeEvent(int)	ILampComponentListener			LAMPTIME-<time>		Lamp time, <time> is elapsed hours
processWarmUpCounterEvent (int)	ILampComponentListener			WARMING-<time>		Warm up counter time, <time> is seconds remaining
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent (byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
getSlide()	ISlideProjectorComponent			?SLIDE		Query for Slide, responds with SLIDE-<slide> where <slide> is 1 to x where x is the maximum support slide (see specific module documentation)

Slide Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
nextSlide()	ISlideProjectorComponent	4			SLIDE_NEXT	Momentary Function Channel: Slide is advanced to the next slide when channel is activated
previousSlide()	ISlideProjectorComponent	5			SLIDE_PREV	Momentary Function Channel: Slide is advanced to the previous slide when channel is activated
setFocusRamp(NEAR)	ISlideProjectorComponent	160			FOCUS_NEAR	Ramping Channel: Focus is ramped near while channel is active
setFocusRamp(FAR)	ISlideProjectorComponent	161			FOCUS_FAR	Ramping Channel: Focus is ramped far while channel is active
setSlide(int)	ISlideProjectorComponent			SLIDE-<slide>		Set Slide where <slide> is 1 to x where x is the maximum support slide (see specific module documentation)
processFocusRampEvent (NEAR)	ISlideProjectorComponentListener	160			FOCUS_NEAR_FB	Feedback Channel: Focus is ramping near while channel is on
processFocusRampEvent (FAR)	ISlideProjectorComponentListener	161			FOCUS_FAR_FB	Feedback Channel: Focus is ramping far while channel is on
processSlideEvent(int)	ISlideProjectorComponentListener			SLIDE-<slide>		Current Slide changed where <slide> is 1 to x where x is the maximum support slide (see specific module documentation)

Slide Projector State Charts

processLampPowerEvent		
State	**Channel 253**	**Channel 255**
OFF	OFF	OFF
COOL	ON	OFF
ON	OFF	ON
WARM	ON	ON

Switcher

- IGainComponent
- IGainComponentListener
- IModuleComponent
- IModuleComponentListener
- IPreAmpComponent
- IPreAmpComponentListener
- ISwitcherComponent
- ISwitcherComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Switcher						
Interface: ISwitcher						
Switcher Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleGainMute()	IGainComponent	144			GAIN_MUTE	Momentary Function Channel: Cycle gain mute when channel is activated
setGain(level)	IGainComponent		5		GAIN_LVL	Set gain, range is 0-255
setGainMuteOn(state)	IGainComponent	143			GAIN_MUTE_ON	Discrete Function Channel: Gain mute is on while channel is active
setGainRamp(UP)	IGainComponent	140			GAIN_UP	Ramping Channel: Gain is ramped up while channel is active
setGainRamp(DOWN)	IGainComponent	141			GAIN_DN	Ramping Channel: Gain is ramped down while channel is active
processGainEvent(int)	IGainComponentListener		5		GAIN_LVL	Gain changed, range is 0-255
processGainMuteEvent (boolean)	IGainComponentListener	143			GAIN_MUTE_FB	Feedback Channel: Gain is muted if channel is on
processGainRampEvent (UP)	IGainComponentListener	140			GAIN_UP_FB	Feedback Channel: Gain is ramping up while channel is on
processGainRampEvent (DOWN)	IGainComponentListener	141			GAIN_DN_FB	Feedback Channel: Gain is ramping down while channel is on
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.

Switcher Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
adjustBalance(1)	IPreAmpComponent	164			BALANCE_UP	Ramping Channel: Balance is incremented when channel is activated
adjustBalance(-1)	IPreAmpComponent	165			BALANCE_DN	Ramping Channel: Balance is decremented when channel is activated
adjustBass(1)	IPreAmpComponent	166			BASS_UP	Ramping Channel: Bass is incremented when channel is activated
adjustBass(-1)	IPreAmpComponent	167			BASS_DN	Ramping Channel: Bass is decremented when channel is activated
adjustEqualizer(band,offset)	IPreAmpComponent			EQUALIZER_OFFSET-(band)=(offset)		Adjust Equalizer Band, given (band) number is a sequential index, and (offset) adjusts its gain, where (offset) ranges from -255 to 255
adjustEqualizer(band[],offset[])	IPreAmpComponent			EQUALIZER_OFFSET-(band)=(offset) [,(band)=(offset)]+		Adjust Equalizer Bands, given each (band) number, with its (offset) to adjust its gain, where (offset)s range from -255 to 255
adjustTreble(1)	IPreAmpComponent	168			TREBLE_UP	Ramping Channel: Treble is incremented when channel is activated
adjustTreble(-1)	IPreAmpComponent	169			TREBLE_DN	Ramping Channel: Treble is decremented when channel is activated
cycleLoudness()	IPreAmpComponent	206			LOUDNESS	Momentary Function Channel: Cycle loudness when channel is activated

Switcher Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getEqualizer()	IPreAmpComponent			?EQUALIZER		Query Equalizer for all supported (bands) and their (gains). Returns EQUALIZER-(band)=(gain)[,(band)=(gain)]+ where (band) ranges from 1 to 255 and its (gain) ranges from 0 to 255
getEqualizer(band)	IPreAmpComponent			?EQUALIZER-(band)		Query Equalizer for the (gain) of a given (band) number, where (band) number ranges from 1 to 255. Returns a format of EQUALIZER-(band)=(gain) where (gain) ranges from 0 to 255
getEqualizer(band[])	IPreAmpComponent			?EQUALIZER-(band)[,(band)]+		Query Equalizer for all given (bands). Returns EQUALIZER-(band)=(gain)[,(band)=(gain)]+ where (band) ranges from 1 to 255 and its corresponding (gain) ranges from 0 to 255
getEqualizerBands()	IPreAmpComponent			?EQUALIZER_BANDS		Query Equalizer for all its supported (bands), where (band) numbers range from 1 to 255. Returns EQUALIZER_BANDS-(band)[,(band)]+
getSurroundMode()	IPreAmpComponent			?SURROUND		Query surround mode, responds with SURROUND-<mode>, where <mode> is MOVIE,MUSIC,OFF
nextSurroundMode()	IPreAmpComponent	170			SURROUND_NEXT	Momentary Function Channel: Next surround mode is selected when channel is activated
previousSurroundMode()	IPreAmpComponent	171			SURROUND_PREV	Momentary Function Channel: Previous surround mode is selected when channel is activated
setBalance(balance)	IPreAmpComponent		2		BALANCE_LVL	Set balance level, range is -128 to 128, -128 is left and 128 is right
setBass(bass)	IPreAmpComponent		3		BASS_LVL	Set bass level, range is 0-255
setEqualizer(band,gain)	IPreAmpComponent			EQUALIZER-(band)=(gain)		Set Equalizer bands gain, given a (band) number and (gain), where (band) ranges from 1 to 255, and (gain) ranges from 0 to 255'
setEqualizer(band[],gain[])	IPreAmpComponent			EQUALIZER-(band)=(gain) [,(band)=(gain)]+		Set Equalizer, given each (band) number which ranges from 1 to 255, and its corresponding (gain), which ranges from 0 to 255
setLoudnessOn(state)	IPreAmpComponent	207			LOUDNESS_ON	Discrete Function Channel: Loudness is on while channel is active
setSurroundMode(sm)	IPreAmpComponent			SURROUND-<mode>		Set surround mode, where <mode> is MOVIE,MUSIC,OFF
setTreble(treble)	IPreAmpComponent		4		TREBLE_LVL	Set treble level, range is 0-255
processBalanceEvent(int)	IPreAmpComponentListener		2		BALANCE_LVL	Balance changed, range is -128 to 128, -128 is left and 128 is right
processBassEvent(int)	IPreAmpComponentListener		3		BASS_LVL	Bass changed, range is 0-255
processEqualizerEvent(int[][])	IPreAmpComponentListener			EQUALIZER-(band)=(gain) [,(band)=(gain)]+		The (gain) changed for the Equalizer at a given (band) number, where (band) ranges from 1 to 255 and its (gain) ranges from 0 to 255
processLoudnessEvent (boolean)	IPreAmpComponentListener	207			LOUDNESS_FB	Feedback Channel: Loudness is on if channel is on
processSurroundModeEvent (SurroundMode)	IPreAmpComponentListener			SURROUND-<mode>		Surround mode changed, where <mode> is MUSIC,MOVIE,OFF
processTrebleEvent(int)	IPreAmpComponentListener		4		TREBLE_LVL	Treble changed, range is 0-255

Switcher Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getInput(sl,output)	ISwitcherComponent			?INPUT-[<sl>,]<output>		Query for the input connected to an output, respond with SWITCH-L<sl>I<input>O<output> where <sl> is ALL, VIDEO, or AUDIO and <input> is 0 if there is no connection. If <sl> is not supplied, ALL will be assumed.
getOutput(sl,input)	ISwitcherComponent			?OUTPUT-[<sl>,]<input>		Query for the outputs connected to an input, respond with SWITCH-L<sl>I<input>O<output>,<output>... where <sl> is ALL, VIDEO, or AUDIO and <output> is 0 if there is no connection. If <sl> is not supplied, ALL will be assumed.
getSwitcherPreset()	ISwitcherComponent			?SWITCHPRESET		Query for switcher preset, responds with SWITCHPRESET-<preset>
saveSwitcherPreset(preset)	ISwitcherComponent			SWITCHPRESETSAVE-<preset>		Save switcher preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setSwitcherPreset(preset)	ISwitcherComponent			SWITCHPRESET-<preset>		Recall switcher preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
switchInputToOutput (AUDIO,input[],output[])	ISwitcherComponent			AI<input,...>O<output, output...>		Switch <input> to one or more <output>s for switcher level Audio. Use <input> 0 for disconnect.
switchInputToOutput (AUDIO,input,output[])	ISwitcherComponent			AI<input>O<output, output...>		Switch <input> to one or more <output>s for switcher level Audio. Use <input> 0 for disconnect.
switchInputToOutput (AUDIO,input,output)	ISwitcherComponent			AI<input>O<output>		Switch <input> to one or more <output>s for switcher level Audio. Use <input> 0 for disconnect.
switchInputToOutput (ALL,input[],output[])	ISwitcherComponent			CI<input,...>O<output, output...>		Switch <input> to one or more <output>s for switcher level All. Use <input> 0 for disconnect.
switchInputToOutput (ALL,input,output[])	ISwitcherComponent			CI<input>O<output, output...>		Switch <input> to one or more <output>s for switcher level All. Use <input> 0 for disconnect.
switchInputToOutput (ALL,input,output)	ISwitcherComponent			CI<input>O<output>		Switch <input> to one or more <output>s for switcher level All. Use <input> 0 for disconnect.
switchInputToOutput (sl,input,output[])	ISwitcherComponent			CL<sl>I<input>O <output, output...>		Switch <input> to one or more <output>s where <sl> is ALL, VIDEO, or AUDIO. Use <input> 0 for disconnect.
switchInputToOutput (sl,input,output)	ISwitcherComponent			CL<sl>I<input>O <output>		Switch <input> to one or more <output>s where <sl> is ALL, VIDEO, or AUDIO. Use <input> 0 for disconnect.
switchInputToOutput (VIDEO,input[],output[])	ISwitcherComponent			VI<input,...>O<output, output...>		Switch <input> to one or more <output>s for switcher level Video. Use <input> 0 for disconnect.
switchInputToOutput (VIDEO,input,output[])	ISwitcherComponent			VI<input>O<output, output...>		Switch <input> to one or more <output>s for switcher level Video. Use <input> 0 for disconnect.
switchInputToOutput (VIDEO,input,output)	ISwitcherComponent			VI<input>O<output>		Switch <input> to one or more <output>s where <sl> is ALL, VIDEO, or AUDIO. Use <input> 0 for disconnect.
processSwitcherPresetEvent (int)	ISwitcherComponentListener			SWITCHPRESET-<preset>		Switcher preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processSwitchEvent (SwitchLevel,int,int)	ISwitcherComponentListener			SWITCH-L<sl>I<input>O <output>		Switch connections changed, where <sl> is ALL, VIDEO, or AUDIO and <input> is 0 if there is no connection.

Switcher Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent (boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent (int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent (UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent (DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Text Keypad

- IKeypadComponent
- IKeypadComponentListener
- IModuleComponent
- IModuleComponentListener
- ITextKeypadComponent
- ITextKeypadComponentListener

Name: Text Keypad						
Interface: ITextKeypad						
Text Keypad Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
addKeypadComponent (index,keypadAddress)	IKeypadComponent			KEYPADADD-<index>, <keypadAddress>		Add a keypad address at a given index, where <index> is 1-x and <keypadaddr> is a keypad address and x is the maximum supported keypad index (see specific module documentation)
getKeypadComponentAddress (index)	IKeypadComponent			?KEYPADADDR-<index>		Query for the address of the keypad at index <index>, responds with KEYPADADDR-<index>,<keypadaddr>
getKeypadComponentIndex (keypadAddress)	IKeypadComponent			?KEYPADIDX-<keypadaddr>		Query for the index of the keypad with address <keypadaddr>, responds with KEYPADADDR-<index>,<keypadaddr>
removeKeypadComponent (keypadAddress)	IKeypadComponent			KEYPADREMOVEADDR-<keypadAddress>		Remove the keypad with address <keypadaddr>, where <keypadaddr> is a keypad address
removeKeypadComponent (index)	IKeypadComponent			KEYPADREMOVEIDX-<index>		Remove the keypad at index <index>, where <index> is 1-x and x is the maximum supported keypad index (see specific module documentation)
setButtonState(btnNum,bs)	IKeypadComponent			KEYPADBTN-<btnAddr>, <state>		Set the state of a keypad button <btn> for the keypad at index/port, where <state> is CLICK or DOUBLECLICK
setButtonStatus(btnNum,bs)	IKeypadComponent	<btn>				Discrete Function Channel: Set the status of a keypad button <btn> for the keypad at index/port (see state chart)
setButtonStatus(btnNum,bs)	IKeypadComponent	<btn>+100				Discrete Function Channel: Set the status of a keypad button <btn> for the keypad at index/port (see state chart)
processButtonStateEvent (PUSH)	IKeypadComponentListener	<btn>				Input Function Channel: Button state is Released when BUTTON_EVENT-RELEASE is received, where <btn> is the button/channel number
processButtonStateEvent (RELEASE)	IKeypadComponentListener	<btn>				Input Function Channel: Button state is Pushed when BUTTON_EVENT-PUSH is received, where <btn> is the button/channel number
processButtonStateEvent (ButtonState)	IKeypadComponentListener			KEYPADBTN-<btn>, <state>		Button State changed. <btn> is button number and <state> is CLICK or DOUBLECLICK
processButtonStatusEvent (ON)	IKeypadComponentListener	<btn>				Feedback Channel: Button status on ON while channel <btn> is on and channel <btn>+100 is off, where <btn> is the button/c channel number (see chart below)

Text Keypad Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processButtonStatusEvent (BLINK)	IKeypadComponentListener	<btn>+100				Feedback Channel: Button status on BLINK while channel <btn> is on and channel <btn>+100 is on, where <btn> is the button/ channel number (see chart below)
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent (byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
clearDisplay()	ITextKeypadComponent			TEXT-		Clear display. "-" is optional
setBacklightOn(state)	ITextKeypadComponent			TEXTBACKLIGHT-<state>		Set BacklightOn where (see specific module documentation)
setText(line,column,text)	ITextKeypadComponent			TEXT-<line>,<column>, <text>		Set text starting at a given line and column. Characters will be overwritten as needed based on text. Text will not wrap around to the next line.

Text Keypad Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setTextDisplay(text)	ITextKeypadComponent			TEXT-<text>		Set text starting at line 1 and column 1. Text will wrap around to the next line to fill the display.
setTextLine(line,text)	ITextKeypadComponent			TEXT-<line>,<text>		Set text starting at column 1 of the line specified. Text will not wrap around to the next line.

Text Keypad State Charts

processButtonStateEvent	
State	**Channel <btn>**
RELEASE	OFF
PUSH	ON

processButtonStatusEvent		
State	**Channel <btn>**	**Channel <btn>+100**
OFF	OFF	OFF
ON	ON	OFF
BLINK	ON	ON
Note: *All Keypad commands include an index. This index is used to obtain the keypad component and the function is called on that component.* *The range for channel '<btn>' is 1 to 100; the corresponding range for channel '<btn> + 100' is 101 to 200.* *Channels above 200 are passed to the module for advanced processing.*		

TV

- IDisplayComponent
- IDisplayComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- ITunerStationComponent
- ITunerStationComponentListener
- ITVComponent
- ITVComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: TV						
Interface: ITV						
TV Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
adjustBrightness(1)	IDisplayComponent	148			BRIGHT_UP	Momentary Function Channel: Brightness is incremented when channel is activated
adjustBrightness(-1)	IDisplayComponent	149			BRIGHT_DN	Momentary Function Channel: Brightness is decremented when channel is activated
adjustColor(1)	IDisplayComponent	150			COLOR_UP	Momentary Function Channel: Color is incremented when channel is activated
adjustColor(-1)	IDisplayComponent	151			COLOR_DN	Momentary Function Channel: Color is dec-remented when channel is activated
adjustContrast(1)	IDisplayComponent	152			CONTRAST_UP	Momentary Function Channel: Contrast is incremented when channel is activated
adjustContrast(-1)	IDisplayComponent	153			CONTRAST_DN	Momentary Function Channel: Contrast is decremented when channel is activated
adjustSharpness(1)	IDisplayComponent	154			SHARP_UP	Momentary Function Channel: Sharpness is incremented when channel is activated
adjustSharpness(-1)	IDisplayComponent	155			SHARP_DN	Momentary Function Channel: Sharpness is decremented when channel is activated
adjustTint(1)	IDisplayComponent	156			TINT_UP	Momentary Function Channel: Tint is incremented when channel is activated

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
adjustTint(-1)	IDisplayComponent	157			TINT_DN	Momentary Function Channel: Tint is decremented when channel is activated
cycleAspectRatio()	IDisplayComponent	142			ASPECT_RATIO	Momentary Function Channel: Cycle aspect ratios when channel is activated
cycleFreeze()	IDisplayComponent	213			PIC_FREEZE	Momentary Function Channel: Cycle freeze when channel is activated
cyclePictureMute()	IDisplayComponent	210			PIC_MUTE	Momentary Function Channel: Cycle picture mute when channel is activated
cyclePIP()	IDisplayComponent	194			PIP	Momentary Function Channel: Cycle PIP when channel is activated
cyclePIPPosition()	IDisplayComponent	191			PIP_POS	Momentary Function Channel: Cycle PIP positions when channel is activated
getActiveWindow()	IDisplayComponent			?ACTIVEWINDOW		Query active window, responds with ACTIVEWINDOW-<window>, where <window> is LEFT,RIGHT,MAIN,SUB
getAspectRatio()	IDisplayComponent			?ASPECT		Query aspect ratio, responds with ASPECT-<ratio>, where <ratio> is ANAMORPHIC,WIDESCREEN,NORMAL
getAspectRatioCount()	IDisplayComponent			?ASPECTRATIO COUNT		Query aspect ratio count, responds with ASPECTRATIOCOUNT-<count>
getAspectRatioProperties(index)	IDisplayComponent			?ASPECTRATIO PROPERTIES		Query properties for all aspect ratios, responds with ASPECTRATIOPROPERTIES-<index>,<displayName>,<value>[;<index>,<displayName>,<value>]*
getAspectRatioProperty(index)	IDisplayComponent			?ASPECTRATIO PROPERTY-<index>		Query properties for single aspect ratio, responds with ASPECTRATIOPROPERTY-<index>,<displayName>,<value>
getAspectRatioSelect()	IDisplayComponent			?ASPECTRATIO SELECT		Gets the index of the currently selected aspect ratio property.
getVideoType()	IDisplayComponent			?VIDEOTYPE		Query video type, responds with VIDEOTYPE-<type> where <type> is AUTO,NTSC,PAL,SECAM
setActiveWindow(mss)	IDisplayComponent			ACTIVEWINDOW-<mss>		Set active window, where <mss> is LEFT,RIGHT,MAIN,SUB
setAspectRatio(aspectRatio)	IDisplayComponent			ASPECT-<aspectRatio>		Set aspect ratio, where <aspectRatio> is ANAMORPHIC,WIDESCREEN,NORMAL
setAspectRatioSelect(index)	IDisplayComponent			ASPECTRATIO SELECT-<index>		Sets the current aspect ratio, where <index> is an integer number between 1 and the value returned by ?ASPECTRATIOSELECT, responds with ASPECTRATIOSELECT-<index>

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setBrightness(level)	IDisplayComponent		10		BRIGHT_LVL	Set brightness level, range is 0-255
setColor(level)	IDisplayComponent		11		COLOR_LVL	Set color level, range is 0-255
setContrast(level)	IDisplayComponent		12		CONTRAST_LVL	Set contrast level, range is 0-255
setFreezeOn(state)	IDisplayComponent	214			PIC_FREEZE_ON	Discrete Function Channel: Freeze is on while channel is active
setPictureMuteOn(state)	IDisplayComponent	211			PIC_MUTE_ON	Discrete Function Channel: Picture Mute is on while channel is active
setPIPOn(state)	IDisplayComponent	195			PIP_ON	Discrete Function Channel: PIP is on while channel is active
setSharpness(level)	IDisplayComponent		13		SHARP_LVL	Set sharpness level, range is 0-255
setTint(level)	IDisplayComponent		14		TINT_LVL	Set tint level, range is 0-255
setVideoType(vt)	IDisplayComponent			VIDEOTYPE-<vt>		Set video type, where <vt> is AUTO,NTSC,PAL,SECAM
swapPIP()	IDisplayComponent	193			PIP_SWAP	Momentary Function Channel: Swap PIP when channel is activated
processActiveWindowEvent (MultiScreenSelect)	IDisplayComponentListener			ACTIVEWINDOW-<window>		Active window changed, where <window> is LEFT,RIGHT,MAIN,SUB
processAspectRatioCountEvent(count)	IDisplayComponentListener			ASPECTRATIO COUNT-<count>		Responds with aspect ratio count, where <count> is an integer value
processAspectRatioEvent(AspectRatio)	IDisplayComponentListener			ASPECT-<ratio>		Aspect ratio changed, where <ratio> is ANAMORPHIC,WIDESCREEN,NORMAL
processAspectRatioPropertiesEvent(String[])	IDisplayComponentListener			ASPECTRATIO PROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>,<value>,...]		Returns properties for each supported aspect ratio, where <index> is an integer, <displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioPropertyEvent(String[])	IDisplayComponentListener			ASPECTRATIOPROPERTY-<index>,<displayName>, <value>		Returns properties for single aspect ratio, where <index> is an integer, <displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioSelectEvent (index)	IDisplayComponentListener			ASPECTRATIOSELECT-<index>		Returns <index> of currently selected aspect ratio.
processBrightnessEvent(int)	IDisplayComponentListener		10		BRIGHT_LVL	Brightness changed, range is 0-255
processColorEvent(int)	IDisplayComponentListener		11		COLOR_LVL	Color changed, range is 0-255
processContrastEvent(int)	IDisplayComponentListener		12		CONTRAST_LVL	Contrast changed, range is 0-255

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processFreezeEvent(boolean)	IDisplayComponentListener	214			PIC_FREEZE_FB	Feedback Channel: Freeze is on if channel is on
processPictureMuteEvent(boolean)	IDisplayComponentListener	211			PIC_MUTE_FB	Feedback Channel: Picture is muted if channel is on
processPIPEvent(boolean)	IDisplayComponentListener	195			PIP_FB	Feedback Channel: PIP is on if channel is on
processSharpnessEvent(int)	IDisplayComponentListener		13		SHARP_LVL	Sharpness changed, range is 0-255
processTintEvent(int)	IDisplayComponentListener		14		TINT_LVL	Tint changed, range is 0-255
processVideoTypeEvent(VideoType)	IDisplayComponentListener			VIDEOTYPE-<type>		Video type changed, where <type> is AUTO,NTSC,PAL,SECAM
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton(RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputPropertiesEvent(String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName> [;<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>]*		Returns comma-separated string containing <index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputPropertyEvent(String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>		Returns comma-separated string containing <index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cycleBand()	ITunerStationComponent	40			TUNER_BAND	Momentary Function Channel: Cycle tuner band when channel is activated
cycleDisplayInfo()	ITunerStationComponent	234			TUNER_OSD	Momentary Function Channel: Cycle on-screen or front-panel display info when channel is activated
cycleStationPresetGroup()	ITunerStationComponent	224			TUNER_PRESET_GROUP	Momentary Function Channel: Cycle station preset group/bank when channel is activated
decrementStation()	ITunerStationComponent	226			TUNER_STATION_DN	Momentary Function Channel: Station is decremented when channel is activated
getBand()	ITunerStationComponent			?BAND		Query for band, responds with BAND-<tb> where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV
getStation()	ITunerStationComponent			?XCH		Query for station, responds with XCH-<station> where <station> is a station string such as "501", "103.7" or "5.1"
getStationPreset()	ITunerStationComponent			?TUNERPRESET		Query for tuner preset, responds with TUNERPRESET-<preset>
getStationPresetCount()	ITunerStationComponent			?STATIONPRESET COUNT		Query for number of valid station presets, responds with STATIONPRESETCOUNT-<count>
getStationPresetProperties()	ITunerStationComponent			?STATIONPRESET PROPERTIES		Query input properties for all station pre-sets, responds with STATIONPRESETPROPERTIES-<index>,<displayName>,<station>[;<index>,<displayName>,<station>]*
getStationPresetProperty(index)	ITunerStationComponent			?STATIONPRESET PROPERTY-<index>		Query properties for a single station preset, responds with STATIONPRESETPROPERTY-<index>,<displayName>,<station>
getStationPresetSelect()	ITunerStationComponent			?STATIONPRESET SELECT		Query for the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
getTunerBandProperties()	ITunerStationComponent			?TUNERBAND PROPERTIES		Query input properties for all tuner bands, responds with TUNERBANDPROPERTIES-<index>, <displayName>,<tb>[,<index>, <displayName>,<tb>]*
getTunerBandProperty(index)	ITunerStationComponent			?TUNERBAND PROPERTY-<index>		Query properties for a single tuner band, responds with TUNERBANDPROPERTY-<index>,<displayName>,<tb>

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getTunerBandPropertyCount()	ITunerStationComponent			?TUNERBAND PROPERTYCOUNT		Query for number of valid tuner bands, responds with TUNERBANDPROPERTYCOUNT-<count>
getTunerBandSelect()	ITunerStationComponent			?TUNERBANDSELECT		Query for the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
getTunerComponentProperties()	ITunerStationComponent			?TUNERCOMPONENT PROPERTIES		Query properties for a single tuner component, responds with TUNERCOMPONENTPROPERTIES-<index>,<displayName>,<value>[;<index>,<displayName>,<value>]*
getTunerComponentProperty()	ITunerStationComponent			?TUNERCOMPONENT PROPERTY		Query input properties for all inputs, responds with TUNERCOMPONENTPROPERTY-<index>,<displayName>,<value>
getTunerComponentPropertyCount()	ITunerStationComponent			?TUNERCOMPONENT PROPERTYCOUNT		Query for number of tuner components, responds with TUNERCOMPONENTPROPERTY-COUNT-<count>
getTunerComponentSelect()	ITunerStationComponent			?TUNERCOMPONENT SELECT		Query for the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
gotoPreviousStation()	ITunerStationComponent	235			TUNER_PREV	Momentary Function Channel: Previous station is selected when channel is activated
incrementStation()	ITunerStationComponent	225			TUNER_STATION_UP	Momentary Function Channel: Station is incremented when channel is activated
nextStationPreset()	ITunerStationComponent	22			CHAN_UP	Momentary Function Channel: Next station preset is selected when channel is activated
previousStationPreset()	ITunerStationComponent	23			CHAN_DN	Momentary Function Channel: Previous station preset is selected when channel is activated
scanStation(FORWARD)	ITunerStationComponent	227			TUNER_SCAN_FWD	Momentary Function Channel: Scans for next station while channel is activate.
scanStation(BACKWARD)	ITunerStationComponent	228			TUNER_SCAN_REV	Momentary Function Channel: Scans for previous station while channel is activate
seekStation(FORWARD)	ITunerStationComponent	229			TUNER_SEEK_FWD	Momentary Function Channel: Seeks for next station while channel is activate

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
seekStation(BACKWARD)	ITunerStationComponent	230			TUNER_SEEK_REV	Momentary Function Channel: Seeks for previous station while channel is activate
setBand(tb)	ITunerStationComponent			BAND-<tb>		Set band, where <tb> is AM,FM, FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV
setStation(station)	ITunerStationComponent			XCH-<station>		Set station, where <station> is a station string such as "501", "103.7" or "5.1"
setStationPreset(preset)	ITunerStationComponent			TUNERPRESET-<preset>		Recall tuner preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setStationPresetSelect(index)	ITunerStationComponent			STATIONPRESET SELECT-<index>		Set the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
setTunerBandSelect(index)	ITunerStationComponent			TUNERBANDSELECT-<index>		Set the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
setTunerComponentSelect(index)	ITunerStationComponent			TUNERCOMPONENT SELECT-<index>		Set the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
processBandEvent(TunerBand)	ITunerStationComponentListener			BAND-<band>		Band changed, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO, LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV
processStationEvent(String)	ITunerStationComponentListener			XCH-<station>		Station changed, where <station> is a station string such as "501", "103.7" or "5.1"
processStationPresetCountEvent(count)	ITunerStationComponentListener			STATIONPRESET COUNT-<count>		Response to ?TUNERCOUNT, where <count> is the integer number of tuner components
processStationPresetEvent(int)	ITunerStationComponentListener			TUNERPRESET-<int>		Tuner preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processStationPresetPropertiesEvent (String[])	ITunerStationComponentListener			STATIONPRESET PROPERTIES-<index>, <displayName>, <value>[;<index>, <displayName>,<value>;...]		Returns the properties for each station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPresetPropertyEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single station preset, where <station> is a string such as "501", "103.7" or "5.1"

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processStationPresetSelectEvent(index)	ITunerStationComponentListener			STATIONPRESETSELECT-<index>		Returns the selected station preset index
processTunerBandPropertiesEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTIES-<index>, <displayName>, <tunerBand>[;<index>, <displayName>, <tunerBand>;...]		Returns properties for each tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandPropertyCountEvent (count)	ITunerStationComponentListener			TUNERBANDPROPERTY COUNT-<count>		Returns the number of valid tuner bands
processTunerBandPropertyEvent(String[])	ITunerStationComponentListener			TUNERBANDPROPERTY-<index>,<displayName>, <tunerBand>		Returns properties for a single tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBandSelectEvent(index)	ITunerStationComponentListener			TUNERBANDSELECT-<index>		Returns the selected tuner band index
processTunerComponentPropertiesEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTIES-<index>, <displayName>, <value>[;<index>, <displayName>,<value>;...]		Returns the properties for each tuner component
processTunerComponentPropertyCountEvent (count)	ITunerStationComponentListener			TUNERCOMPONENT PROPERTYCOUNT		Returns the integer number of tuner components
processTunerComponentPropertyEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single tuner component
processTunerComponentSelectEvent(index)	ITunerStationComponentListener			TUNERCOMPONENT SELECT-<index>		Returns the selected tuner component index
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active

TV Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

TV State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

UPS

- IModuleComponent
- IModuleComponentListener
- IUPSComponent
- IUPSComponentListener

Name: UPS						
Interface: IUPS						
UPS Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

UPS Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getAlarms()	IUPSComponent			?UPSALARMS		Query for alarms, responds with UPSALARMS-<alarms> where <alarms> is a CSV of alarm names
getBackupProperty (backupProperty)	IUPSComponent			?UPSBACKUP PROPERTY-<property>		Query for the <bu-prop> property, responds with UPSBACKUPPROPERTY-<bu-prop>,<value>
getBackupStatus()	IUPSComponent			?UPSBACKUP STATUS		Query for backup status of UPS, responds with UPSBACKUPSTATUS
getBackupTimer(backupTimer)	IUPSComponent			?UPSBACKUP TIMER-<timer>		Query for <bu-timer> value, responds with UPSBACKUPTIMER-<bu-timer>,<value>
getInputProperty(inputProperty)	IUPSComponent			?UPSINPUTPROPERTY-<property>		Query for <input-prop>, responds with UPSINPUTPROPERTY-<input-prop>,<value>
getOutletCount()	IUPSComponent			?UPSOUTLETCOUNT		Query for number of outlets
getOutletProperties (outletNumber)	IUPSComponent			?UPSOUTLET PROPERTIES-<outletNumber>		Query the <outlet#> for all available properties, responds with UPSOUTLETPROPERTIES [<outlet#>,<outlet-prop>,<value>] for each property, CSV format
getOutletProperty (outletNumber,outletProperty)	IUPSComponent			?UPSOUTLET PROPERTY-<outletNumber>, <outletNumber>		Query the <outlet#> for the <outlet-prop>, responds with UPSOUTLETPROPERTY-<outlet#>,<outlet-prop>,<value>
getOutletState(outlet)	IUPSComponent			?UPSOUTLETSTATE-<outletNumber>		Query for the state of the specified outlet, responds with UPSOUTLETSTATE
getState()	IUPSComponent			?UPSSTATE		Gets the current state of the UPS, responds with UPSSTATE
getStatus()	IUPSComponent			?UPSSTATUS		Query for status of UPS, responds with UPSSTATUS
getTemperatureScale()	IUPSComponent			?UPSTEMPERATURE SCALE		Query for the temperature scale, responds with UPSTEMPERATURESCALE-<tempscale>
getTestResult()	IUPSComponent			?UPSTESTRESULT		Query for the results of the last UPSTEST-<test>,<START>, responds with UPSTESTRESULT-<test>,<PASS/FAIL>
getTestsSupported()	IUPSComponent			?UPSTEST SUPPORTED		Query for supported tests, responds with UPSTESTSSUPPORTED-<test_list> where <test_list> is a CSV of supported tests
setOutletState(outlet,state)	IUPSComponent			UPSOUTLETSTATE-<outletNumber>, <state>		Sets the outlets state, where <outlet#> is the integer outlet number and <outlet-state> is ON or OFF
setOutletState(1,state)	IUPSComponent			UPSOUTLETSTATE-1, <state>		Sets the outlets 1 state, where <outlet-state> is ON or OFF
setOutletState(2,state)	IUPSComponent			UPSOUTLETSTATE-2, <state>		Sets the outlets 2 state, where <outlet-state> is ON or OFF
setOutletState(3,state)	IUPSComponent			UPSOUTLETSTATE-3, <state>		Sets the outlets 3 state, where <outlet-state> is ON or OFF

UPS Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setOutletState(4,state)	IUPSComponent			UPSOUTLETSTATE-4, <state>		Sets the outlets 4 state, where <outlet-state> is ON or OFF
setOutletState(5,state)	IUPSComponent			UPSOUTLETSTATE-5, <state>		Sets the outlets 5 state, where <outlet-state> is ON or OFF
setOutletState(6,state)	IUPSComponent			UPSOUTLETSTATE-6, <state>		Sets the outlets 6 state, where <outlet-state> is ON or OFF
setState(state)	IUPSComponent			UPSSTATE-<state>		set UPS to the given state
setState(state,delay)	IUPSComponent			UPSSTATE-<state>, <delay>		set UPS to the given state in <delay> seconds
setTemperatureScale (temperatureScale)	IUPSComponent			UPSTEMPERATURE SCALE- <temperatureScale>		Sets the temperature scale
setTest(test,testAction)	IUPSComponent			UPSTEST-<test>, <testAction>		Start/Stop/Interrupt the specified <test>, responds with UPSTEST-<test>,<START/STOP/INTERRUPT>
processAlarmsEvent (arg,alarms)	IUPSComponentListener			UPSALARMS-<alarms>		<alarms> is a CSV string containing the alarm names
processBackupPropertyEvent (UPSBackupProperties,string)	IUPSComponentListener			UPSBACKUP PROPERTY- <bu-prop>,<value>		Data is UPSBackupProperties, <bu-prop> is TSE values as String in upper case (CHARGE, VOLTAGE, CURRENT, REPLACEDATE, or TEMPERATURE)
processBackupStatusEvent (UPSBackupStatus)	IUPSComponentListener			UPSBACKUPSTATUS- <bu-status>		Data is UPSBackupStatus, <bu-status> is TSE values as String in upper case (OK, LOW, or DEPLETED)
processBackupTimerEvent (UPSBackupTimer, backupTimerValue)	IUPSComponentListener			UPSBACKUPTIMER- <bu-timer>,<value>		Data is UPSBackupTimer, <bu-timer> is TSE values as String in upper case (TIMEONBATTERY, TIMETODEPLETED, or TIMETOLOW), <value> is seconds
processInputPropertyEvent (UPSInputProperties, inputPropertyValue)	IUPSComponentListener			UPSINPUT PROPERTY- <input-prop>,<value>		Data is UPSInputProperties, <input-prop> is TSE values as String in upper case (DELAYONTIME, DELAYOFFTIME, LINES, FREQUENCY, VOLTAGE, CURRENT, or CAPACITY)
processIsBackupSupportedEvent (BackupSupported)	IUPSComponentListener			UPSBACKUP SUPPORTED- <boolean>		Response to ?UPSBACKUPSUPPORTED
processOutletCountEvent (outletCount)	IUPSComponentListener			UPSOUTLETCOUNT- <count>		Response to ?UPSOUTLETCOUNTT, where <count> is the integer number of outlets
processOutletPropertiesEvent (outletNumber, outletPropertyValues)	IUPSComponentListener			UPSOUTLET PROPERTIES- <outletNumber>, <outlet-prop>,<value>[, <outletNumber>, <outlet-prop>,<value>,...]		Data is UPSOutletProperties, <outlet-prop> is TSE values as String in upper case (OUTPUTLINES, OUTPUTFREQUENCY, OUTPUTVOLTAGE, OUTPUTCURRENT, OUTPUTPOWER, OUTPUTPOWERSOURCE, OUTPUTLOAD, BYPASSLINES, BYPASSFREQUENCY, BYPASSVOLTAGE, BYPASSCURRENT, BYPASSPOWER, BYPASSPOWERSOURCE, or BYPASSLOAD)

UPS Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processOutletPropertyEvent (outletNumber, UPSOutletProperties, outletPropertyValue)	IUPSComponentListener			UPSOUTLET PROPERTY-<outletNumber>, <outlet-prop>,<value>		Data is UPSOutletProperties, <outlet-prop> is TSE values as String in upper case (OUTPUTLINES, OUTPUTFREQUENCY, OUTPUTVOLTAGE, OUTPUTCURRENT, OUTPUTPOWER, OUTPUTPOWERSOURCE, OUTPUTLOAD, BYPASSLINES, BYPASSFREQUENCY, BYPASSVOLTAGE, BYPASSCURRENT, BYPASSPOWER, BYPASSPOWERSOURCE, or BYPASSLOAD)
processOutletStateEvent (UPSOutletState,outletNumber)	IUPSComponentListener			UPSOUTLETSTATE-<outletNumber>, <state>		Response to ?UPSOUTLETSTATE-<outlet#>,<outlet-state>, where <outlet-state> is ON or OFF
processOutletStateEvent (UPSOutletState,1)	IUPSComponentListener			UPSOUTLETSTATE-1, <state>		Response to ?OUTLETSTATE-1
processOutletStateEvent (UPSOutletState,2)	IUPSComponentListener			UPSOUTLETSTATE-2, <state>		Response to ?OUTLETSTATE-2
processOutletStateEvent (UPSOutletState,3)	IUPSComponentListener			UPSOUTLETSTATE-3, <state>		Response to ?OUTLETSTATE-3
processOutletStateEvent (UPSOutletState,4)	IUPSComponentListener			UPSOUTLETSTATE-4, <state>		Response to ?OUTLETSTATE-4
processOutletStateEvent (UPSOutletState,5)	IUPSComponentListener			UPSOUTLETSTATE-5, <state>		Response to ?OUTLETSTATE-5
processOutletStateEvent (UPSOutletState,6)	IUPSComponentListener			UPSOUTLETSTATE-6, <state>		Response to ?OUTLETSTATE-6
processStateEvent (UPSState,delay)	IUPSComponentListener			UPSSTATE-<state>, <delay>		Data is UPSState, <ups_state> is TSE values as String in upper case (SHUTDOWN, REBOOT, OUTPUT_ONLY, STARTUP, NORMAL, BATTERY, BYPASS, MANUAL_BYPASS, REDUCING, or BOOSTING)
processStatusEvent(UPSState)	IUPSComponentListener			UPSSTATUS-<status>		Data is UPSStatus, <ups_status> is TSE values as String in upper case (SHUTDOWN, REBOOT, OUTPUT_ONLY, STARTUP, NORMAL, BATTERY, BYPASS, MANUAL_BYPASS, REDUCING, or BOOSTING)
processTemperatureScaleEvent (TemeratureScale)	IUPSComponentListener			UPSTEMPERATURE SCALE-<temperatureScale>		Data is TemperatureScale, <tempscale> is TSE values as String in upper case (CELSIUS, OR FAHRENHEIT)
processTestEvent(TestAction)	IUPSComponentListener			UPSTEST-<test>, <action>		Data is TestAction, <action> is TSE values as String in upper case (START, STOP or INTERRUPT)
processTestResultEvent (TestResult)	IUPSComponentListener			UPSTESTRESULT-<test>,<result>		Data is TestResult, <result> is TSE values as String in upper case (PASS or FAIL)
processTestsSupportedEvent (arg,tests)	IUPSComponentListener			UPSTESTS SUPPORTED-<test_list>		<test_list> is a CSV string containing each <test> supported

Utility

- IModuleComponent
- IModuleComponentListener
- IUtilityComponent
- IUtilityComponentListener

Name: Utility						
Interface: IUtility						
Utility Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy><hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.

VCR

- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- ITapeTransportComponent
- ITapeTransportComponentListener
- ITunerStationComponent
- ITunerStationComponentListener
- IVCRComponent
- IVCRComponentListener

Name: VCR						
Interface: IVCR						
VCR Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton (THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton (UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton (PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton (SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton (INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>, <signalType>,<deviceLabel>,<displayName> [;<index>,<inputGroup>,<signalType>,<deviceLabel>, <displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>, <signalType>,<deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputProperties Event(String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>[; <index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>]*		Returns comma-separated string containing <index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputProperty Event(String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>, <displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cycleSearchSpeed()	ITapeTransportComponent	119			SEARCH_SPEED	Momentary Function Channel: Cycle search speed when channel is activated
ejectTape()	ITapeTransportComponent	120			EJECT	Momentary Function Channel: Tape is ejected when channel is activated
resetTapeCounter()	ITapeTransportComponent	121			RESET_COUNTER	Momentary Function Channel: Counter is reset when channel is activated
setTapeCounterNotificationOn (state)	ITapeTransportComponent			TAPECOUNTER NOTIFY-<state>		Turn counter notification on or off, where <state> is 1 or 0
setTapeTransport(PLAY)	ITapeTransportComponent	1			PLAY	Momentary Function Channel: Deck is set to play when the channel is activated
setTapeTransport(STOP)	ITapeTransportComponent	2			STOP	Momentary Function Channel: Deck is set to stop when the channel is activated
setTapeTransport(PAUSE)	ITapeTransportComponent	3			PAUSE	Momentary Function Channel: Deck is set to pause when the channel is activated
setTapeTransport(FF)	ITapeTransportComponent	4			FFWD	Momentary Function Channel: Deck is set to fast-forward when the channel is activated
setTapeTransport(REW)	ITapeTransportComponent	5			REW	Momentary Function Channel: Deck is set to rewind when the channel is activated
setTapeTransport(SEARCH_FWD)	ITapeTransportComponent	6			SFWD	Momentary Function Channel: Deck is set to search forward when the channel is activated
setTapeTransport(SEARCH_REV)	ITapeTransportComponent	7			SREV	Momentary Function Channel: Deck is set to search reverse when the channel is activated
setTapeTransport(RECORD)	ITapeTransportComponent	8			RECORD	Momentary Function Channel: Deck is set to record when the channel is activated

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setTapeTransport(SLOW_FWD)	ITapeTransportComponent	188			SLOW_FWD	Momentary Function Channel: Deck is set to slow forward when the channel is activated
setTapeTransport(SLOW_REV)	ITapeTransportComponent	189			SLOW_REV	Momentary Function Channel: Deck is set to slow reverse when the channel is activated
processTapeCounterEvent (MediaTime)	ITapeTransportComponentListener			TAPECOUNTER-<counter>		Tape counter changed, <counter> is hh:mm:ss.ff
processTapeLoadedEvent (boolean)	ITapeTransportComponentListener	122			TAPE_LOADED_FB	Feedback Channel: Tape is loaded while channel is on
processTapeRecordLockedEvent (boolean)	ITapeTransportComponentListener	123			RECORD_LOCK_FB	Feedback Channel: Tape is record locked while channel is on
processTapeTransportEvent (PLAY)	ITapeTransportComponentListener	241			PLAY_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (STOP)	ITapeTransportComponentListener	242			STOP_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (PAUSE)	ITapeTransportComponentListener	243			PAUSE_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (FF)	ITapeTransportComponentListener	244			FFWD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (REW)	ITapeTransportComponentListener	245			REW_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SEARCH_FWD)	ITapeTransportComponentListener	246			SFWD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SEARCH_REV)	ITapeTransportComponentListener	247			SREV_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (RECORD)	ITapeTransportComponentListener	248			RECORD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SLOW_FWD)	ITapeTransportComponentListener	249			SLOW_FWD_FB	Feedback Channel: Transport state change (see chart below)
processTapeTransportEvent (SLOW_REV)	ITapeTransportComponentListener	250			SLOW_REV_FB	Feedback Channel: Transport state change (see chart below)
cycleBand()	ITunerStationComponent	40			TUNER_BAND	Momentary Function Channel: Cycle tuner band when channel is activated
cycleDisplayInfo()	ITunerStationComponent	234			TUNER_OSD	Momentary Function Channel: Cycle on-screen or front-panel display info when channel is activated
cycleStationPresetGroup()	ITunerStationComponent	224			TUNER_PRESET_GROUP	Momentary Function Channel: Cycle station preset group/bank when channel is activated
decrementStation()	ITunerStationComponent	226			TUNER_STATION_DN	Momentary Function Channel: Station is decremented when channel is activated
getBand()	ITunerStationComponent			?BAND		Query for band, responds with BAND-<tb> where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getStation()	ITunerStationComponent			?XCH		Query for station, responds with XCH-<station> where <station> is a station string such as "501", "103.7" or "5.1"
getStationPreset()	ITunerStationComponent			?TUNERPRESET		Query for tuner preset, responds with TUNERPRESET-<preset>
getStationPresetCount()	ITunerStationComponent			?STATIONPRESET COUNT		Query for number of valid station presets, responds with STATIONPRESETCOUNT-<count>
getStationPresetProperties()	ITunerStationComponent			?STATIONPRESET PROPERTIES		Query input properties for all station presets, responds with STATIONPRESETPROPERTIES-<index>, <displayName>,<station>[;<index>,<displayName>, <station>]*
getStationPresetProperty(index)	ITunerStationComponent			?STATIONPRESET PROPERTY-<index>		Query properties for a single station preset, responds with STATIONPRESETPROPERTY-<index>, <displayName>,<station>
getStationPresetSelect()	ITunerStationComponent			?STATIONPRESET SELECT		Query for the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
getTunerBandProperties()	ITunerStationComponent			?TUNERBAND PROPERTIES		Query input properties for all tuner bands, responds with TUNERBANDPROPERTIES-<index>,<displayName>, <tb>[,<index>,<displayName>,<tb>]*
getTunerBandProperty(index)	ITunerStationComponent			?TUNERBAND PROPERTY-<index>		Query properties for a single tuner band, responds with TUNERBANDPROPERTY-<index>,<displayName>,<tb>
getTunerBandPropertyCount()	ITunerStationComponent			?TUNERBAND PROPERTYCOUNT		Query for number of valid tuner bands, responds with TUNERBANDPROPERTYCOUNT-<count>
getTunerBandSelect()	ITunerStationComponent			?TUNERBANDSELECT		Query for the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
getTunerComponentProperties()	ITunerStationComponent			?TUNERCOMPONENT PROPERTIES		Query properties for a single tuner component, responds with TUNERCOMPONENTPROPERTIES-<index>, <displayName>,<value>[;<index>,<displayName>, <value>]*
getTunerComponentProperty()	ITunerStationComponent			?TUNERCOMPONENT PROPERTY		Query input properties for all inputs, responds with TUNERCOMPONENTPROPERTY-<index>, <displayName>,<value>
getTunerComponent PropertyCount()	ITunerStationComponent			?TUNERCOMPONENT PROPERTYCOUNT		Query for number of tuner components, responds with TUNERCOMPONENTPROPERTYCOUNT-<count>
getTunerComponentSelect()	ITunerStationComponent			?TUNERCOMPONENT SELECT		Query for the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
gotoPreviousStation()	ITunerStationComponent	235			TUNER_PREV	Momentary Function Channel: Previous station is selected when channel is activated
incrementStation()	ITunerStationComponent	225			TUNER_STATION_UP	Momentary Function Channel: Station is incremented when channel is activated

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
nextStationPreset()	ITunerStationComponent	22			CHAN_UP	Momentary Function Channel: Next station preset is selected when channel is activated
previousStationPreset()	ITunerStationComponent	23			CHAN_DN	Momentary Function Channel: Previous station preset is selected when channel is activated
scanStation(FORWARD)	ITunerStationComponent	227			TUNER_SCAN_FWD	Momentary Function Channel: Scans for next station while channel is activate.
scanStation(BACKWARD)	ITunerStationComponent	228			TUNER_SCAN_REV	Momentary Function Channel: Scans for previous station while channel is activate
seekStation(FORWARD)	ITunerStationComponent	229			TUNER_SEEK_FWD	Momentary Function Channel: Seeks for next station while channel is activate
seekStation(BACKWARD)	ITunerStationComponent	230			TUNER_SEEK_REV	Momentary Function Channel: Seeks for previous station while channel is activate
setBand(tb)	ITunerStationComponent			BAND-<tb>		Set band, where <tb> is AM,FM,FM_MONO, SATELLITE_RADIO,LONG_WAVE,MEDIUM_WAVE, SHORT_WAVE,TV
setStation(station)	ITunerStationComponent			XCH-<station>		Set station, where <station> is a station string such as "501", "103.7" or "5.1"
setStationPreset(preset)	ITunerStationComponent			TUNERPRESET-<preset>		Recall tuner preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setStationPresetSelect(index)	ITunerStationComponent			STATIONPRESET SELECT-<index>		Set the currently selected index of valid station presets, responds with STATIONPRESETSELECT-<index>
setTunerBandSelect(index)	ITunerStationComponent			TUNERBANDSELECT-<index>		Set the currently selected index of valid tuner bands, responds with TUNERBANDSELECT-<index>
setTunerComponentSelect(index)	ITunerStationComponent			TUNERCOMPONENT SELECT-<index>		Set the currently selected index of tuner components, responds with TUNERCOMPONENTSELECT-<index>
processBandEvent(TunerBand)	ITunerStationComponentListener			BAND-<band>		Band changed, where <tb> is AM,FM,FM_MONO,SATELLITE_RADIO,LONG_WAVE, MEDIUM_WAVE,SHORT_WAVE,TV
processStationEvent(String)	ITunerStationComponentListener			XCH-<station>		Station changed, where <station> is a station string such as "501", "103.7" or "5.1"
processStationPreset CountEvent(count)	ITunerStationComponentListener			STATIONPRESET COUNT-<count>		Response to ?TUNERCOUNT, where <count> is the integer number of tuner components
processStationPreset Event(int)	ITunerStationComponentListener			TUNERPRESET-<int>		Tuner preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processStationPreset PropertiesEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTIES-<index>, <displayName>, <value>[;<index>, <displayName>, <value>;...]		Returns the properties for each station preset, where <station> is a string such as "501", "103.7" or "5.1"

VCR Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processStationPreset PropertyEvent(String[])	ITunerStationComponentListener			STATIONPRESET PROPERTY-<index>, <displayName>,<value>		Returns the properties for a single station preset, where <station> is a string such as "501", "103.7" or "5.1"
processStationPreset SelectEvent(index)	ITunerStationComponentListener			STATIONPRESET SELECT-<index>		Returns the selected station preset index
processTunerBand PropertiesEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTIES-<index>,<displayName>, <tunerBand>[;<index>, <displayName>, <tunerBand>;...]		Returns properties for each tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBand PropertyCountEvent(count)	ITunerStationComponentListener			TUNERBAND PROPERTYCOUNT-<count>		Returns the number of valid tuner bands
processTunerBand PropertyEvent(String[])	ITunerStationComponentListener			TUNERBAND PROPERTY-<index>,<displayName>, <tunerBand>		Returns properties for a single tuner band, where <tb> is AM, FM, FM_MONO, SATELLITE_RADIO, LONG_WAVE, MEDIUM_WAVE, SHORT_WAVE, TV
processTunerBand SelectEvent(index)	ITunerStationComponentListener			TUNERBANDSELECT-<index>		Returns the selected tuner band index
processTunerComponent PropertiesEvent(String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTIES-<index>,<displayName>, <value>[;<index>, <displayName>, <value>;...]		Returns the properties for each tuner component
processTunerComponent PropertyCountEvent(count)	ITunerStationComponentListener			TUNERCOMPONENT PROPERTYCOUNT		Returns the integer number of tuner components
processTunerComponent PropertyEvent (String[])	ITunerStationComponentListener			TUNERCOMPONENT PROPERTY-<index>, <displayName>, <value>		Returns the properties for a single tuner component
processTunerComponent SelectEvent(index)	ITunerStationComponentListener			TUNERCOMPONENT SELECT-<index>		Returns the selected tuner component index

VCR State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

processTapeTransportEvent										
State	**Channel 241**	**Channel 242**	**Channel 243**	**Channel 244**	**Channel 245**	**Channel 246**	**Channel 247**	**Channel 248**	**Channel 249**	**Channel 250**
PLAY	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
STOP	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
FF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
REW	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
SEARCH_FWD	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
SEARCH_REV	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
RECORD	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
RECORD_PAUSE	OFF	OFF	ON	OFF	OFF	OFF	OFF	ON	OFF	OFF
SLOW_FWD	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
SLOW_REV	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Video Conferencer

- ICameraComponent
- ICameraComponentListener
- IDialerComponent
- IDialerComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPhonebookComponent
- IPhonebookComponentListener
- IPowerComponent
- IPowerComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- IVideoConferencerComponent
- IVideoConferencerComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Video Conferencer

Interface: IVideoConferencer

Video Conferencer Functions

Name	Component	Channel	Level	Command	Constant	Notes
cycleAutoFocus()	ICameraComponent	172			AUTO_FOCUS	Momentary Function Channel: Cycle auto focus when channel is activated
cycleAutoIris()	ICameraComponent	173			AUTO_IRIS	Momentary Function Channel: Cycle auto iris when channel is activated
cycleCameraPreset()	ICameraComponent	177			CAM_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getCameraComponentCount()	ICameraComponent			?CAMERACOMPONENT COUNT		Query number of camera components, responds with CAMERACOMPONENT-COUNT-<count>
getCameraPreset()	ICameraComponent			?CAMERAPRESET		Query for camera preset, responds with CAMERAPRESET-<preset>
getCameraPresetCount()	ICameraComponent			?CAMERAPRESET COUNT		Query number of presets on a camera, responds with CAMERAPRESETCOUNT-<count>
getCameraPresetProperties()	ICameraComponent			?CAMERAPRESET PROPERTIES		Query properties for every camera preset, responds with CAMERAPRESETPROPERTIES-<index>,<displayName>[;<index>,<displayName>]

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getCameraPresetProperty(index)	ICameraComponent			?CAMERAPRESET PROPERTY		Query properties for a single camera preset, responds with CAMERAPRESETPROPERTY-<index>, <displayName>
saveCameraPreset(preset)	ICameraComponent			CAMERAPRESET SAVE-<preset>		Save Camera Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setAutoFocusOn(state)	ICameraComponent	162			AUTO_FOCUS_ON	Discrete Function Channel: Auto focus is on while channel is active
setAutoIrisOn(state)	ICameraComponent	163			AUTO_IRIS_ON	Discrete Function Channel: Auto iris is on while channel is active
setCameraPreset(preset)	ICameraComponent			CAMERAPRESET-<preset>		Recall camera preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
setFocus(focus)	ICameraComponent		16		FOCUS_LVL	Set focus position, range is 0-255 (0=near)
setFocusRamp(NEAR)	ICameraComponent	160			FOCUS_NEAR	Ramping Channel: Focus is ramped near while channel is active
setFocusRamp(FAR)	ICameraComponent	161			FOCUS_FAR	Ramping Channel: Focus is ramped far while channel is active
setFocusSpeed(speed)	ICameraComponent		19		FOCUS_SPEED_LVL	Set focus speed, range is 0-255 (0=slow)
setIris(iris)	ICameraComponent		17		IRIS_LVL	Set iris position, range is 0-255 (0=closed)
setIrisRamp(OPEN)	ICameraComponent	174			IRIS_OPEN	Ramping Channel: Iris is ramped open while channel is active
setIrisRamp(CLOSE)	ICameraComponent	175			IRIS_CLOSE	Ramping Channel: Iris is ramped closed while channel is active
setIrisSpeed(speed)	ICameraComponent		20		IRIS_SPEED_LVL	Set iris speed, range is 0-255 (0=slow)
setPan(pan)	ICameraComponent		27		PAN_LVL	Set pan position, range is 0-255 (0=left)
setPanRamp(LEFT)	ICameraComponent	134			PAN_LT	Ramping Channel: Pan is ramped left while channel is active
setPanRamp(RIGHT)	ICameraComponent	135			PAN_RT	Ramping Channel: Pan is ramped right while channel is active
setPanSpeed(speed)	ICameraComponent		29		PAN_SPEED_LVL	Set pan speed, range is 0-255 (0=slow)
setTilt(tilt)	ICameraComponent		28		TILT_LVL	Set tilt position, range is 0-255 (0=down)
setTiltRamp(UP)	ICameraComponent	132			TILT_UP	Ramping Channel: Tilt is ramped up while channel is active
setTiltRamp(DOWN)	ICameraComponent	133			TILT_DN	Ramping Channel: Tilt is ramped down while channel is active

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setTiltSpeed(speed)	ICameraComponent		30		TILT_SPEED_LVL	Set tilt speed, range is 0-255 (0=slow)
setZoom(zoom)	ICameraComponent		15		ZOOM_LVL	Set zoom position, range is 0-255 (0=out/ Wide)
setZoomRamp(OUT)	ICameraComponent	158			ZOOM_OUT	Ramping Channel: Zoom is ramped out(wide) while channel is active
setZoomRamp(IN)	ICameraComponent	159			ZOOM_IN	Ramping Channel: Zoom is ramped in (tele) while channel is active
setZoomSpeed(speed)	ICameraComponent		18		ZOOM_SPEED_LVL	Set zoom speed, range is 0-255 (0=slow)
processAutoFocusEvent(boolean)	ICameraComponentListener	162			AUTO_FOCUS_FB	Feedback Channel: Auto focus is on if channel is active
processAutoIrisEvent(boolean)	ICameraComponentListener	163			AUTO_IRIS_FB	Feedback Channel: Auto iris is on if channel is active
processCameraComponentCountEvent (count)	ICameraComponentListener			CAMERACOMPONENT COUNT-<count>		Returns integer number of camera components
processCameraPresetCountEvent (count)	ICameraComponentListener			CAMERAPRESET COUNT-<count>		Returns integer number of camera presets
processCameraPresetEvent(int)	ICameraComponentListener			CAMERAPRESET-<preset>		Camera preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processCameraPresetPropertiesEvent (String[])	ICameraComponentListener			CAMERAPRESET PROPERTIES-<index>, <displayName> [;-<index>, <displayName>,]*		Returns string containing <index> and <displayName> for each camera preset.
processCameraPresetPropertyEvent (String[])	ICameraComponentListener			CAMERAPRESET PROPERTY-<index>, <displayName>		Returns string containing <index> and <displayName> for a camera preset.
processFocusEvent(int)	ICameraComponentListener		16		FOCUS_LVL	Focus changed, range is 0-255 (0=near)
processFocusRampEvent(NEAR)	ICameraComponentListener	160			FOCUS_NEAR_FB	Feedback Channel: Focus is ramping near while channel is on
processFocusRampEvent(FAR)	ICameraComponentListener	161			FOCUS_FAR_FB	Feedback Channel: Focus is ramping far while channel is on
processFocusSpeedEvent(int)	ICameraComponentListener		19		FOCUS_SPEED_LVL	Focus speed changed, range is 0-255 (0=slow)
processIrisEvent(int)	ICameraComponentListener		17		IRIS_LVL	Iris changed, range is 0-255 (0=closed)
processIrisRampEvent(OPEN)	ICameraComponentListener	174			IRIS_OPEN_FB	Feedback Channel: Iris is ramping open while channel is on
processIrisRampEvent(CLOSE)	ICameraComponentListener	175			IRIS_CLOSE_FB	Feedback Channel: Iris is ramping closed while channel is on
processIrisSpeedEvent(int)	ICameraComponentListener		20		IRIS_SPEED_LVL	Iris speed changed, range is 0-255 (0=slow)

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPanEvent(int)	ICameraComponentListener		27		PAN_LVL	Pan changed, range is 0-255 (0=left)
processPanRampEvent(LEFT)	ICameraComponentListener	134			PAN_LT_FB	Feedback Channel: Pan is ramping left while channel is on
processPanRampEvent(RIGHT)	ICameraComponentListener	135			PAN_RT_FB	Feedback Channel: Pan is ramping right up while channel is on
processPanSpeedEvent(int)	ICameraComponentListener		29		PAN_SPEED_LVL	Pan speed changed, range is 0-255 (0=slow)
processTiltEvent(int)	ICameraComponentListener		28		TILT_LVL	Tilt changed, range is 0-255 (0=down)
processTiltRampEvent(UP)	ICameraComponentListener	132			TILT_UP_FB	Feedback Channel: Tilt is ramping up while channel is on
processTiltRampEvent(DOWN)	ICameraComponentListener	133			TILT_DN_FB	Feedback Channel: Tilt is ramping down while channel is on
processTiltSpeedEvent(int)	ICameraComponentListener		30		TILT_SPEED_LVL	Tilt speed changed, range is 0-255 (0=slow)
processZoomEvent(int)	ICameraComponentListener		15		ZOOM_LVL	Zoom changed, range is 0-255 (0=out/wide)
processZoomRampEvent(OUT)	ICameraComponentListener	158			ZOOM_OUT_FB	Feedback Channel: Zoom is ramping out (wide) while channel is on
processZoomRampEvent(IN)	ICameraComponentListener	159			ZOOM_IN_FB	Feedback Channel: Zoom is ramping in (tele) while channel is on
processZoomSpeedEvent(int)	ICameraComponentListener		18		ZOOM_SPEED_LVL	Zoom speed changed, range is 0-255 (0=slow)
cycleAudibleRing()	IDialerComponent	205			DIAL_AUDIBLE_RING	Momentary Function Channel: Cycle audible ring when channel is activated
cycleAutoAnswer()	IDialerComponent	204			DIAL_AUTO_ANSWER	Momentary Function Channel: Cycle auto answer when channel is activated
cycleOffHook()	IDialerComponent	202			DIAL_OFF_HOOK	Momentary Function Channel: Cycle hook state when channel is activated
dial(recordID)	IDialerComponent			DIALID-<recordID>		Dial a speed dial record, where <id> is a record id returned by a speed dial search.
dial(index)	IDialerComponent			DIALINDEX-<index>		Dial a speed dial index, where <index> is 1 to x and x is the maximum supported speed dial index (see specific module documentation)
dialDTMF(char)	IDialerComponent			DTMF-<digit>		Send a DTMF tone for a character without regard for hook status
dialNumber(number)	IDialerComponent			DIALNUMBER-<number>		Dial a number where <number> is the number to be dialed.
flashHook()	IDialerComponent	208			DIAL_FLASH_HOOK	Momentary Function Channel: Flash hook when channel is activated

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDialerStatus()	IDialerComponent			?DIALERSTATUS		Query dialer status, responds with DIALER-STATUS-<status>, where <status> is DIALING,BUSY,RINGING,DISCONNECTED, NEGOTIATING,FAULT,CONNECTED
redial()	IDialerComponent	201			DIAL_REDIAL	Momentary Function Channel: Redial when channel is activated
setAudibleRingOn(state)	IDialerComponent	240			DIAL_AUDIBLE_RING_ON	Discrete Function Channel: Audible ring is on while channel is active
setAutoAnswerOn(state)	IDialerComponent	239			DIAL_AUTO_ANSWER_ON	Discrete Function Channel: Auto answer is on while channel is active
setOffHook(state)	IDialerComponent	238			DIAL_OFF_HOOK_ON	Discrete Function Channel: Hook state is off hook while channel is active
processAudibleRingOnEvent (boolean)	IDialerComponentListener	240			DIAL_AUDIBLE_RING_FB	Feedback Channel: Audible ring is on if channel is on
processAutoAnswerOnEvent (boolean)	IDialerComponentListener	239			DIAL_AUTO_ANSWER_FB	Feedback Channel: Auto answer is on if channel is on
processDialerStatusEvent(BUSY)	IDialerComponentListener			DIALERSTATUS-BUSY		Dialer status changed, number being dialed is busy
processDialerStatusEvent (CONNECTED)	IDialerComponentListener			DIALERSTATUS-CONNECTED		Dialer status changed, dialer is connected
processDialerStatusEvent (DIALING)	IDialerComponentListener			DIALERSTATUS-DIALING		Dialer status changed, dialer is dialing
processDialerStatusEvent (DISCONNECTED)	IDialerComponentListener			DIALERSTATUS-DISCONNECTED		Dialer status changed, dialer is disconnected/ idle
processDialerStatusEvent(FAULT)	IDialerComponentListener			DIALERSTATUS-FAULT		Dialer status changed, dialer encounter a fault during dialing/negotiating
processDialerStatusEvent (NEGOTIATING)	IDialerComponentListener			DIALERSTATUS-NEGOTIATING		Dialer status changed, dialer is negotiating
processDialerStatusEvent (RINGING)	IDialerComponentListener			DIALERSTATUS-RINGING		Dialer status changed, number being dialed is ringing
processIncomingCallEvent (String)	IDialerComponentListener			INCOMINGCALL-<number>		An incoming call is detected. If available via caller ID, the phone number will be supplied
processOffHookEvent(boolean)	IDialerComponentListener	238			DIAL_OFF_HOOK_FB	Feedback Channel: Hook state is off hook if channel is on
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton(THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton(UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton(PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton(SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton(INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
closeSearchDB(searchHandle)	IPhonebookComponent			PHONEBOOK CLOSESEARCH-<searchHandle>		Close the search associated with search key = <key>
deleteRecord(recordID)	IPhonebookComponent			PHONEBOOKDELETE-<recordID>		Delete phonebook record with index/id of <id>

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getPhonebookCapacity()	IPhonebookComponent			?PHONEBOOK CAPACITY		Query for the phonebook capacity. Responds with PHONEBOOKCAPACITY-<count> where count is 1 to x and x is the maximum supported phonebook index (see specific module documentation)
queryDB(si)	IPhonebookComponent			PHONEBOOKNEXT-<searchHandle> [,<count>,<position>]		Request next <count> records starting with record <position> for the phonebook search associated with search key <key>. If <count> is not present, it is assumed to be 1. If <position> is not present, it is assumed to be the next record in the search result set based on the last PHONEBOOKNEXT or PHONEBOOKPREV command. Responses with PHONEBOOKNEXT-<key>,<count>, <position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
queryDB(si)	IPhonebookComponent			PHONEBOOK PREV-<searchHandle> [,<count>]		Request previous <count> records for the phonebook search associated with search key <key>. If <count> is not present, it is assumed to be 1. Responses with PHONEBOOKPREV-<key>,<count>, <position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
queryDB(si)	IPhonebookComponent			PHONEBOOK REFRESH-<searchHandle> [,<count>]		Refresh <count> records starting with the last starting position used with a PHONEBOOKNEXT or PHONEBOOKPREV command for the phonebook search associated with search key <key>. If <count> is not present, it is assumed to be 1. Responses with PHONEBOOKNEXT-<key>,<count>,<position> where <key> is the search key, <count> is the number of records to expect and <position> is the position of the first record to be returned between 1 through the total number of records.
searchDB(sr)	IPhonebookComponent			PHONEBOOKSEARCH-<searchHandle>,ID=<id>		Search the phonebook database for records with ID equal to <id>. If <id> is "*", all records are returned. <key> is a search key used in other search operations, such as closeSearchDB(). It can be any string you like, such as a panel device number or internal key that makes sense for your program. All future DB operation associated with this search will reference this key value.

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
updateRecord(sdr)	IPhonebookComponent			PHONEBOOKUPDATE-<recordID>,<name>,<number>		Update phonebook record with index/id of <id>, where <name> is the new name and <number> is the new number
processCloseEvent(String)	IPhonebookComponentListener			PHONEBOOK CLOSESEARCH-<searchHandle>		The search associated with search key <key> have been closed. No more operations can be performed against this search.
processDeleteRecordEvent (boolean,String)	IPhonebookComponentListener			PHONEBOOKDELETE-<recordID>,<success>		Phonebook record with index/id of <id> was deleted if <success> is 1, otherwise delete failed.
processEndOfSetEvent(String)	IPhonebookComponentListener			PHONEBOOKEND-<searchHandle>		End of search set was reached for search with search key <key>
processQueryDBEvent (PhonebookRecord)	IPhonebookComponentListener			PHONEBOOKRECORD-<searchHandle>,<recordID>,<resultNumber>,<name>,<number>		Phonebook record for search with search key <key>. <id> is the record ID, <#> is the record position from 1 to the total number of records, <name> is the name/label and <number> is the phone number.
processSearchDBEvent (PhonebookSearchSet)	IPhonebookComponentListener			PHONEBOOK SEARCHRESULT-<searchHandle>,<count>		Phonebook search results are available for search with search key <key>. <count> is the total count of records. Use PHONEBOOKNEXT and PHONEBOOKPREV to get more records.
processStartOfSetEvent(String)	IPhonebookComponentListener			PHONEBOOKSTART-<searchHandle>		Beginning of search set was reached for search with search key <key>
processUpdateRecordEvent (boolean,String)	IPhonebookComponentListener			PHONEBOOKUPDATE-<recordID>,<success>		Phonebook record with index/id of <id> was updated if <success> is 1, otherwise update failed.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputPropertiesEvent(String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processInputPropertyEvent(String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cyclePIP()	IVideoConferencerComponent	194			PIP	Momentary Function Channel: Cycle PIP when channel is activated
cyclePIPPosition()	IVideoConferencerComponent	191			PIP_POS	Momentary Function Channel: Cycle PIP positions when channel is activated
cyclePrivacy()	IVideoConferencerComponent	145			VCONF_PRIVACY	Momentary Function Channel: Cycle privacy when channel is activated
setPIPOn(state)	IVideoConferencerComponent	195			PIP_ON	Discrete Function Channel: PIP is on while channel is active
setPrivacyOn(state)	IVideoConferencerComponent	146			VCONF_PRIVACY_ON	Discrete Function Channel: Privacy is on while channel is active
swapPIP()	IVideoConferencerComponent	193			PIP_SWAP	Momentary Function Channel: Swap PIP when channel is activated
train()	IVideoConferencerComponent	147			VCONF_TRAIN	Momentary Function Channel: Train is executed when the channel is activated
processPIPEvent(boolean)	IVideoConferencerComponentListener	195			PIP_FB	Feedback Channel: PIP is on if channel is on
processPrivacyEvent(boolean)	IVideoConferencerComponentListener	146			VCONF_PRIVACY_FB	Feedback Channel: Privacy is muted if channel is on
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active

Video Conferencer Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent(DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Video Conferencer State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Video Processor

- IDisplayComponent
- IDisplayComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- IVideoProcessorComponent
- IVideoProcessorComponentListener

Name: Video Processor						
Interface: IVideoProcessor						
Video Processor Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
adjustBrightness(1)	IDisplayComponent	148			BRIGHT_UP	Momentary Function Channel: Brightness is incremented when channel is activated
adjustBrightness(-1)	IDisplayComponent	149			BRIGHT_DN	Momentary Function Channel: Brightness is decremented when channel is activated
adjustColor(1)	IDisplayComponent	150			COLOR_UP	Momentary Function Channel: Color is incremented when channel is activated
adjustColor(-1)	IDisplayComponent	151			COLOR_DN	Momentary Function Channel: Color is decremented when channel is activated
adjustContrast(1)	IDisplayComponent	152			CONTRAST_UP	Momentary Function Channel: Contrast is incremented when channel is activated
adjustContrast(-1)	IDisplayComponent	153			CONTRAST_DN	Momentary Function Channel: Contrast is decremented when channel is activated
adjustSharpness(1)	IDisplayComponent	154			SHARP_UP	Momentary Function Channel: Sharpness is incremented when channel is activated
adjustSharpness(-1)	IDisplayComponent	155			SHARP_DN	Momentary Function Channel: Sharpness is decremented when channel is activated
adjustTint(1)	IDisplayComponent	156			TINT_UP	Momentary Function Channel: Tint is incremented when channel is activated
adjustTint(-1)	IDisplayComponent	157			TINT_DN	Momentary Function Channel: Tint is decremented when channel is activated
cycleAspectRatio()	IDisplayComponent	142			ASPECT_RATIO	Momentary Function Channel: Cycle aspect ratios when channel is activated

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleFreeze()	IDisplayComponent	213			PIC_FREEZE	Momentary Function Channel: Cycle freeze when channel is activated
cyclePictureMute()	IDisplayComponent	210			PIC_MUTE	Momentary Function Channel: Cycle picture mute when channel is activated
cyclePIP()	IDisplayComponent	194			PIP	Momentary Function Channel: Cycle PIP when channel is activated
cyclePIPPosition()	IDisplayComponent	191			PIP_POS	Momentary Function Channel: Cycle PIP positions when channel is activated
getActiveWindow()	IDisplayComponent			?ACTIVEWINDOW		Query active window, responds with ACTIVEWINDOW-<window>, where <window> is LEFT,RIGHT,MAIN,SUB
getAspectRatio()	IDisplayComponent			?ASPECT		Query aspect ratio, responds with ASPECT-<ratio>, where <ratio> is ANAMORPHIC, WIDESCREEN,NORMAL
getAspectRatioCount()	IDisplayComponent			?ASPECTRATIOCOUNT		Query aspect ratio count, responds with ASPECTRATIOCOUNT-<count>
getAspectRatioProperties(index)	IDisplayComponent			?ASPECTRATIO PROPERTIES		Query properties for all aspect ratios, responds with ASPECTRATIOPROPERTIES-<index>, <displayName>,<value>[;<index>, <displayName>,<value>]*
getAspectRatioProperty(index)	IDisplayComponent			?ASPECTRATIOPROPERTY-<index>		Query properties for single aspect ratio, responds with ASPECTRATIOPROPERTY-<index>, <displayName>,<value>
getAspectRatioSelect()	IDisplayComponent			?ASPECTRATIOSELECT		Gets the index of the currently selected aspect ratio property.
getVideoType()	IDisplayComponent			?VIDEOTYPE		Query video type, responds with VIDEOTYPE-<type> where <type> is AUTO,NTSC,PAL, SECAM
setActiveWindow(mss)	IDisplayComponent			ACTIVEWINDOW-<mss>		Set active window, where <mss> is LEFT,RIGHT, MAIN,SUB
setAspectRatio(aspectRatio)	IDisplayComponent			ASPECT-<aspectRatio>		Set aspect ratio, where <aspectRatio> is ANAMORPHIC,WIDESCREEN,NORMAL
setAspectRatioSelect(index)	IDisplayComponent			ASPECTRATIOSELECT-<index>		Sets the current aspect ratio, where <index> is an integer number between 1 and the value returned by ?ASPECTRATIOSELECT, responds with ASPECTRATIOSELECT-<index>
setBrightness(level)	IDisplayComponent		10		BRIGHT_LVL	Set brightness level, range is 0-255
setColor(level)	IDisplayComponent		11		COLOR_LVL	Set color level, range is 0-255
setContrast(level)	IDisplayComponent		12		CONTRAST_LVL	Set contrast level, range is 0-255

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setFreezeOn(state)	IDisplayComponent	214			PIC_FREEZE_ON	Discrete Function Channel: Freeze is on while channel is active
setPictureMuteOn(state)	IDisplayComponent	211			PIC_MUTE_ON	Discrete Function Channel: Picture Mute is on while channel is active
setPIPOn(state)	IDisplayComponent	195			PIP_ON	Discrete Function Channel: PIP is on while channel is active
setSharpness(level)	IDisplayComponent		13		SHARP_LVL	Set sharpness level, range is 0-255
setTint(level)	IDisplayComponent		14		TINT_LVL	Set tint level, range is 0-255
setVideoType(vt)	IDisplayComponent			VIDEOTYPE-<vt>		Set video type, where <vt> is AUTO,NTSC,PAL,SECAM
swapPIP()	IDisplayComponent	193			PIP_SWAP	Momentary Function Channel: Swap PIP when channel is activated
processActiveWindowEvent (MultiScreenSelect)	IDisplayComponentListener			ACTIVEWINDOW-<window>		Active window changed, where <window> is LEFT,RIGHT,MAIN,SUB
processAspectRatioCountEvent (count)	IDisplayComponentListener			ASPECTRATIOCOUNT-<count>		Responds with aspect ratio count, where <count> is an integer value
processAspectRatioEvent (AspectRatio)	IDisplayComponentListener			ASPECT-<ratio>		Aspect ratio changed, where <ratio> is ANAMORPHIC,WIDESCREEN,NORMAL
processAspectRatioPropertiesEvent (String[])	IDisplayComponentListener			ASPECTRATIO PROPERTIES-<index>, <displayName>,<value> [;<index>,<displayName>, <value>,...]		Returns properties for each supported aspect ratio, where <index> is an integer, <displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioPropertyEvent (String[])	IDisplayComponentListener			ASPECTRATIOPROPERTY-<index>,<displayName>, <value>		Returns properties for single aspect ratio, where <index> is an integer,<displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioSelectEvent (index)	IDisplayComponentListener			ASPECTRATIOSELECT-<index>		Returns <index> of currently selected aspect ratio.
processBrightnessEvent(int)	IDisplayComponentListener		10		BRIGHT_LVL	Brightness changed, range is 0-255
processColorEvent(int)	IDisplayComponentListener		11		COLOR_LVL	Color changed, range is 0-255
processContrastEvent(int)	IDisplayComponentListener		12		CONTRAST_LVL	Contrast changed, range is 0-255
processFreezeEvent(boolean)	IDisplayComponentListener	214			PIC_FREEZE_FB	Feedback Channel: Freeze is on if channel is on
processPictureMuteEvent(boolean)	IDisplayComponentListener	211			PIC_MUTE_FB	Feedback Channel: Picture is muted if channel is on
processPIPEvent(boolean)	IDisplayComponentListener	195			PIP_FB	Feedback Channel: PIP is on if channel is on
processSharpnessEvent(int)	IDisplayComponentListener		13		SHARP_LVL	Sharpness changed, range is 0-255
processTintEvent(int)	IDisplayComponentListener		14		TINT_LVL	Tint changed, range is 0-255

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processVideoTypeEvent (VideoType)	IDisplayComponentListener			VIDEOTYPE-<type>		Video type changed, where <type> is AUTO, NTSC,PAL,SECAM
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO
pressMenuButton (THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN
pressMenuButton (UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton (SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton (INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>,<value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nld,nIOChan)	IPowerComponent			IOLINK-<nld>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>[;<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>,<signalType>,<deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.

Video Processor Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputPropertiesEvent (String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>, <signalType>,<deviceLabel>, <displayName>[;<index>, <inputGroup>,<signalType>, <deviceLabel>, <displayName>]*		Returns comma-separated string containing <index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputPropertyEvent(String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>, <inputGroup>,<signalType>, <deviceLabel>, <displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
cycleVideoProcessorPreset()	IVideoProcessorComponent	137			VPROC_PRESET	Momentary Function Channel: Cycle video processor preset when channel is activated
getVideoProcessorPreset()	IVideoProcessorComponent			?VPROCPRESET		Query for video processor preset, responds with VPROCPRESET-<preset>
saveVideoProcessorPreset (nPresetNum)	IVideoProcessorComponent			VPROCPRESETSAVE-<preset>		Save video processor Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVideoProcessorPreset (nPresetNum)	IVideoProcessorComponent			VPROCPRESET-<preset>		Recall video processor preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
processVideoProcessorPresetEvent (int)	IVideoProcessorComponentListener			VPROCPRESET-<preset>		Video processor preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)

Video Processor State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Video Projector

- IDisplayComponent
- IDisplayComponentListener
- ILampComponent
- ILampComponentListener
- IMenuComponent
- IMenuComponentListener
- IModuleComponent
- IModuleComponentListener
- ISourceSelectComponent
- ISourceSelectComponentListener
- IVideoProjectorComponent
- IVideoProjectorComponentListener
- IVolumeComponent
- IVolumeComponentListener

Name: Video Projector						
Interface: IVideoProjector						
Video Projector Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
adjustBrightness(1)	IDisplayComponent	148			BRIGHT_UP	Momentary Function Channel: Brightness is i incremented when channel is activated
adjustBrightness(-1)	IDisplayComponent	149			BRIGHT_DN	Momentary Function Channel: Brightness is decremented when channel is activated
adjustColor(1)	IDisplayComponent	150			COLOR_UP	Momentary Function Channel: Color is incremented when channel is activated
adjustColor(-1)	IDisplayComponent	151			COLOR_DN	Momentary Function Channel: Color is decremented when channel is activated
adjustContrast(1)	IDisplayComponent	152			CONTRAST_UP	Momentary Function Channel: Contrast is incremented when channel is activated
adjustContrast(-1)	IDisplayComponent	153			CONTRAST_DN	Momentary Function Channel: Contrast is decremented when channel is activated
adjustSharpness(1)	IDisplayComponent	154			SHARP_UP	Momentary Function Channel: Sharpness is incremented when channel is activated
adjustSharpness(-1)	IDisplayComponent	155			SHARP_DN	Momentary Function Channel: Sharpness is decremented when channel is activated
adjustTint(1)	IDisplayComponent	156			TINT_UP	Momentary Function Channel: Tint is incremented when channel is activated
adjustTint(-1)	IDisplayComponent	157			TINT_DN	Momentary Function Channel: Tint is decremented when channel is activated

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
cycleAspectRatio()	IDisplayComponent	142			ASPECT_RATIO	Momentary Function Channel: Cycle aspect ratios when channel is activated
cycleFreeze()	IDisplayComponent	213			PIC_FREEZE	Momentary Function Channel: Cycle freeze when channel is activated
cyclePictureMute()	IDisplayComponent	210			PIC_MUTE	Momentary Function Channel: Cycle picture mute when channel is activated
cyclePIP()	IDisplayComponent	194			PIP	Momentary Function Channel: Cycle PIP when channel is activated
cyclePIPPosition()	IDisplayComponent	191			PIP_POS	Momentary Function Channel: Cycle PIP positions when channel is activated
getActiveWindow()	IDisplayComponent			?ACTIVEWINDOW		Query active window, responds with ACTIVEWINDOW-<window>, where <window> is LEFT,RIGHT,MAIN,SUB
getAspectRatio()	IDisplayComponent			?ASPECT		Query aspect ratio, responds with ASPECT-<ratio>, where <ratio> is ANAMORPHIC,WIDESCREEN, NORMAL
getAspectRatioCount()	IDisplayComponent			?ASPECTRATIOCOUNT		Query aspect ratio count, responds with ASPECTRATIOCOUNT-<count>
getAspectRatioProperties(index)	IDisplayComponent			?ASPECTRATIO PROPERTIES		Query properties for all aspect ratios, responds with ASPECTRATIOPROPERTIES-<index>, <displayName>,<value>[;<index>,<displayName>, <value>]*
getAspectRatioProperty(index)	IDisplayComponent			?ASPECTRATIO PROPERTY-<index>		Query properties for single aspect ratio, responds with ASPECTRATIOPROPERTY-<index>,<displayName>, <value>
getAspectRatioSelect()	IDisplayComponent			?ASPECTRATIO SELECT		Gets the index of the currently selected aspect ratio property.
getVideoType()	IDisplayComponent			?VIDEOTYPE		Query video type, responds with VIDEOTYPE-<type> where <type> is AUTO,NTSC,PAL,SECAM
setActiveWindow(mss)	IDisplayComponent			ACTIVEWINDOW-<mss>		Set active window, where <mss> is LEFT,RIGHT, MAIN,SUB
setAspectRatio(aspectRatio)	IDisplayComponent			ASPECT-<aspectRatio>		Set aspect ratio, where <aspectRatio> is ANAMORPHIC,WIDESCREEN,NORMAL
setAspectRatioSelect(index)	IDisplayComponent			ASPECTRATIOSELECT-<index>		Sets the current aspect ratio, where <index> is an integer number between 1 and the value returned by ?ASPECTRATIOSELECT, responds with ASPECTRATIOSELECT-<index>
setBrightness(level)	IDisplayComponent		10		BRIGHT_LVL	Set brightness level, range is 0-255
setColor(level)	IDisplayComponent		11		COLOR_LVL	Set color level, range is 0-255
setContrast(level)	IDisplayComponent		12		CONTRAST_LVL	Set contrast level, range is 0-255

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setFreezeOn(state)	IDisplayComponent	214			PIC_FREEZE_ON	Discrete Function Channel: Freeze is on while channel is active
setPictureMuteOn(state)	IDisplayComponent	211			PIC_MUTE_ON	Discrete Function Channel: Picture Mute is on while channel is active
setPIPOn(state)	IDisplayComponent	195			PIP_ON	Discrete Function Channel: PIP is on while channel is active
setSharpness(level)	IDisplayComponent		13		SHARP_LVL	Set sharpness level, range is 0-255
setTint(level)	IDisplayComponent		14		TINT_LVL	Set tint level, range is 0-255
setVideoType(vt)	IDisplayComponent			VIDEOTYPE-<vt>		Set video type, where <vt> is AUTO,NTSC,PAL, SECAM
swapPIP()	IDisplayComponent	193			PIP_SWAP	Momentary Function Channel: Swap PIP when channel is activated
processActiveWindowEvent (MultiScreenSelect)	IDisplayComponentListener			ACTIVEWINDOW-<window>		Active window changed, where <window> is LEFT, RIGHT,MAIN,SUB
processAspectRatioCountEvent (count)	IDisplayComponentListener			ASPECTRATIOCOUNT-<count>		Responds with aspect ratio count, where <count> is an integer value
processAspectRatioEvent (AspectRatio)	IDisplayComponentListener			ASPECT-<ratio>		Aspect ratio changed, where <ratio> is ANAMORPHIC, WIDESCREEN,NORMAL
processAspectRatioPropertiesEvent (String[])	IDisplayComponentListener			ASPECTRATIO PROPERTIES-<index>, <displayName>, <value>[;<index>, <displayName>, <value>,...]		Returns properties for each supported aspect ratio, where <index> is an integer, <displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioPropertyEvent (String[])	IDisplayComponentListener			ASPECTRATIO PROPERTY-<index>, <displayName>, <value>		Returns properties for single aspect ratio, where <index> is an integer,<displayName> is the device text and <value> is NORMAL, WIDESCREEN, ANAMORPHIC or unique to that device
processAspectRatioSelectEvent (index)	IDisplayComponentListener			ASPECTRATIO SELECT-<index>		Returns <index> of currently selected aspect ratio.
processBrightnessEvent(int)	IDisplayComponentListener		10		BRIGHT_LVL	Brightness changed, range is 0-255
processColorEvent(int)	IDisplayComponentListener		11		COLOR_LVL	Color changed, range is 0-255
processContrastEvent(int)	IDisplayComponentListener		12		CONTRAST_LVL	Contrast changed, range is 0-255
processFreezeEvent(boolean)	IDisplayComponentListener	214			PIC_FREEZE_FB	Feedback Channel: Freeze is on if channel is on
processPictureMuteEvent(boolean)	IDisplayComponentListener	211			PIC_MUTE_FB	Feedback Channel: Picture is muted if channel is on
processPIPEvent(boolean)	IDisplayComponentListener	195			PIP_FB	Feedback Channel: PIP is on if channel is on
processSharpnessEvent(int)	IDisplayComponentListener		13		SHARP_LVL	Sharpness changed, range is 0-255
processTintEvent(int)	IDisplayComponentListener		14		TINT_LVL	Tint changed, range is 0-255

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processVideoTypeEvent (VideoType)	IDisplayComponentListener			VIDEOTYPE-<type>		Video type changed, where <type> is AUTO,NTSC,PAL,SECAM
cycleLampPower()	ILampComponent	9			POWER	Momentary Function Channel: Cycle lamp power when channel is activated
getCoolDownTime()	ILampComponent			?COOLDOWN		Query for cool down time, responds with COOLDOWN-<time> where <time> is in seconds
getLampTime()	ILampComponent			?LAMPTIME		Query for lamp time, responds with LAMPTIME-<time> where <time> is in hours
getWarmUpTime()	ILampComponent			?WARMUP		Query for warm up time, responds with WARMUP-<time> where <time> is in seconds
setCoolDownTime(secs)	ILampComponent			COOLDOWN-<seconds>		Set cool down time where <time> is in seconds
setCounterNotificationOn(state)	ILampComponent			COUNTERNOTIFY-<state>		Turn counter notification on or off, where <state> is 1 or 0
setLampPower(ON)	ILampComponent	27			PWR_ON	Momentary Function Channel: Lamp power is turned on when channel is activated
setLampPower(OFF)	ILampComponent	28			PWR_OFF	Momentary Function Channel: Lamp power is turned off when channel is activated
setLampPower(ps)	ILampComponent	255			POWER_ON	Discrete Function Channel: Lamp power is on while channel is active
setLampTime(hours)	ILampComponent			LAMPTIME-<hours>		Set lamp time where <time> is in hours
setWarmUpTime(secs)	ILampComponent			WARMUP-<seconds>		Set warm up time where <time> is in seconds
processCoolDownCounterEvent (int)	ILampComponentListener			COOLING-<time>		Cool down counter time, <time> is seconds remaining
processLampPowerEvent (WARMING_UP)	ILampComponentListener	253			LAMP_WARMING_FB	Feedback Channel: Indicates Lamp is warming and cannot accept commands
processLampPowerEvent (COOLING_DOWN)	ILampComponentListener	254			LAMP_COOLING_FB	Feedback Channel: Indicates Lamp is cooling and cannot accept commands
processLampPowerEvent(ON)	ILampComponentListener	255			LAMP_POWER_FB	Feedback Channel: Indicates lamp is ON
processLampTimeEvent(int)	ILampComponentListener			LAMPTIME-<time>		Lamp time, <time> is elapsed hours
processWarmUpCounterEvent(int)	ILampComponentListener			WARMING-<time>		Warm up counter time, <time> is seconds remaining
moveMenuCursor(UP)	IMenuComponent	45			MENU_UP	Momentary Function Channel: Move menu cursor UP
moveMenuCursor(DOWN)	IMenuComponent	46			MENU_DN	Momentary Function Channel: Move menu cursor DOWN
moveMenuCursor(LEFT)	IMenuComponent	47			MENU_LT	Momentary Function Channel: Move menu cursor LEFT
moveMenuCursor(RIGHT)	IMenuComponent	48			MENU_RT	Momentary Function Channel: Move menu cursor RIGHT

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
moveMenuCursor(UP_LEFT)	IMenuComponent	51			MENU_UP_LT	Momentary Function Channel: Move menu cursor UP_LEFT
moveMenuCursor(UP_RIGHT)	IMenuComponent	52			MENU_UP_RT	Momentary Function Channel: Move menu cursor UP_RIGHT
moveMenuCursor(DOWN_LEFT)	IMenuComponent	53			MENU_DN_LT	Momentary Function Channel: Move menu cursor DOWN_LEFT
moveMenuCursor(DOWN_RIGHT)	IMenuComponent	54			MENU_DN_RT	Momentary Function Channel: Move menu cursor DOWN_RIGHT
pressMenuButton(DIGIT_0)	IMenuComponent	10			DIGIT_0	Momentary Function Channel: Press menu button DIGIT_0
pressMenuButton(DIGIT_1)	IMenuComponent	11			DIGIT_1	Momentary Function Channel: Press menu button DIGIT_1
pressMenuButton(DIGIT_2)	IMenuComponent	12			DIGIT_2	Momentary Function Channel: Press menu button DIGIT_2
pressMenuButton(DIGIT_3)	IMenuComponent	13			DIGIT_3	Momentary Function Channel: Press menu button DIGIT_3
pressMenuButton(DIGIT_4)	IMenuComponent	14			DIGIT_4	Momentary Function Channel: Press menu button DIGIT_4
pressMenuButton(DIGIT_5)	IMenuComponent	15			DIGIT_5	Momentary Function Channel: Press menu button DIGIT_5
pressMenuButton(DIGIT_6)	IMenuComponent	16			DIGIT_6	Momentary Function Channel: Press menu button DIGIT_6
pressMenuButton(DIGIT_7)	IMenuComponent	17			DIGIT_7	Momentary Function Channel: Press menu button DIGIT_7
pressMenuButton(DIGIT_8)	IMenuComponent	18			DIGIT_8	Momentary Function Channel: Press menu button DIGIT_8
pressMenuButton(DIGIT_9)	IMenuComponent	19			DIGIT_9	Momentary Function Channel: Press menu button DIGIT_9
pressMenuButton(PLUS_10)	IMenuComponent	20			MENU_PLUS_10	Momentary Function Channel: Press menu button PLUS_10
pressMenuButton(ENTER)	IMenuComponent	21			MENU_ENTER	Momentary Function Channel: Press menu button ENTER
pressMenuButton(CANCEL)	IMenuComponent	43			MENU_CANCEL	Momentary Function Channel: Press menu button CANCEL
pressMenuButton(MENU)	IMenuComponent	44			MENU_FUNC	Momentary Function Channel: Press menu button MENU
pressMenuButton(EXIT)	IMenuComponent	50			MENU_EXIT	Momentary Function Channel: Press menu button EXIT
pressMenuButton(VIDEO)	IMenuComponent	57			MENU_VIDEO	Momentary Function Channel: Press menu button VIDEO

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (THUMBS_DOWN)	IMenuComponent	58			MENU_THUMBS_DN	Momentary Function Channel: Press menu button THUMBS_DOWN
pressMenuButton(THUMBS_UP)	IMenuComponent	59			MENU_THUMBS_UP	Momentary Function Channel: Press menu button THUMBS_UP
pressMenuButton(ACCEPT)	IMenuComponent	60			MENU_ACCEPT	Momentary Function Channel: Press menu button ACCEPT to answer an incoming call
pressMenuButton(REJECT)	IMenuComponent	61			MENU_REJECT	Momentary Function Channel: Press menu button REJECT to reject an incoming call
pressMenuButton(LIVE_TV)	IMenuComponent	62			MENU_LIVE_TV	Momentary Function Channel: Press menu button LIVE_TV
pressMenuButton(SLEEP)	IMenuComponent	63			MENU_SLEEP	Momentary Function Channel: Press menu button SLEEP
pressMenuButton(PPV)	IMenuComponent	64			MENU_PPV	Momentary Function Channel: Press menu button PPV
pressMenuButton(FUNCTION)	IMenuComponent	65			MENU_FUNCTION	Momentary Function Channel: Press menu button FUNCTION
pressMenuButton(SETUP)	IMenuComponent	66			MENU_SETUP	Momentary Function Channel: Press menu button SETUP
pressMenuButton(XM)	IMenuComponent	77			MENU_XM	Momentary Function Channel: Press menu button XM
pressMenuButton(FM)	IMenuComponent	78			MENU_FM	Momentary Function Channel: Press menu button FM
pressMenuButton(AM)	IMenuComponent	79			MENU_AM	Momentary Function Channel: Press menu button AM
pressMenuButton(CLEAR)	IMenuComponent	80			MENU_CLEAR	Momentary Function Channel: Press menu button CLEAR
pressMenuButton(BACK)	IMenuComponent	81			MENU_BACK	Momentary Function Channel: Press menu button BACK
pressMenuButton(FORWARD)	IMenuComponent	82			MENU_FORWARD	Momentary Function Channel: Press menu button FORWARD
pressMenuButton(ADVANCE)	IMenuComponent	83			MENU_ADVANCE	Momentary Function Channel: Press menu button ADVANCE
pressMenuButton(DIMMER)	IMenuComponent	84			MENU_DIMMER	Momentary Function Channel: Press menu button DIMMER
pressMenuButton(HOLD)	IMenuComponent	85			MENU_HOLD	Momentary Function Channel: Press menu button HOLD
pressMenuButton(LIST)	IMenuComponent	86			MENU_LIST	Momentary Function Channel: Press menu button LIST
pressMenuButton(LEFT_PAREN)	IMenuComponent	87			MENU_LT_PAREN	Momentary Function Channel: Press menu button LEFT_PAREN
pressMenuButton(RIGHT_PAREN)	IMenuComponent	88			MENU_RT_PAREN	Momentary Function Channel: Press menu button RIGHT_PAREN

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (UNDER_SCORE)	IMenuComponent	89			MENU_UNDERSCORE	Momentary Function Channel: Press menu button UNDER_SCORE
pressMenuButton(DASH)	IMenuComponent	90			MENU_DASH	Momentary Function Channel: Press menu button DASH
pressMenuButton(ASTERISK)	IMenuComponent	91			MENU_ASTERISK	Momentary Function Channel: Press menu button ASTERISK
pressMenuButton(DOT)	IMenuComponent	92			MENU_DOT	Momentary Function Channel: Press menu button DOT
pressMenuButton(POUND)	IMenuComponent	93			MENU_POUND	Momentary Function Channel: Press menu button POUND
pressMenuButton(COMMA)	IMenuComponent	94			MENU_COMMA	Momentary Function Channel: Press menu button COMMA
pressMenuButton(DIAL)	IMenuComponent	95			MENU_DIAL	Momentary Function Channel: Press menu button DIAL
pressMenuButton(CONFERENCE)	IMenuComponent	96			MENU_CONFERENCE	Momentary Function Channel: Press menu button CONFERENCE
pressMenuButton(PLUS_100)	IMenuComponent	97			MENU_PLUS_100	Momentary Function Channel: Press menu button PLUS_100
pressMenuButton(PLUS_1000)	IMenuComponent	98			MENU_PLUS_1000	Momentary Function Channel: Press menu button PLUS_1000
pressMenuButton(DISPLAY)	IMenuComponent	99			MENU_DISPLAY	Momentary Function Channel: Press menu button DISPLAY
pressMenuButton(SUBTITLE)	IMenuComponent	100			MENU_SUBTITLE	Momentary Function Channel: Press menu button SUBTITLE
pressMenuButton(INFO)	IMenuComponent	101			MENU_INFO	Momentary Function Channel: Press menu button INFO
pressMenuButton(FAVORITES)	IMenuComponent	102			MENU_FAVORITES	Momentary Function Channel: Press menu button FAVORITES
pressMenuButton(CONTINUE)	IMenuComponent	103			MENU_CONTINUE	Momentary Function Channel: Press menu button CONTINUE
pressMenuButton(RETURN)	IMenuComponent	104			MENU_RETURN	Momentary Function Channel: Press menu button RETURN
pressMenuButton(GUIDE)	IMenuComponent	105			MENU_GUIDE	Momentary Function Channel: Press menu button GUIDE
pressMenuButton(PAGE_UP)	IMenuComponent	106			MENU_PAGE_UP	Momentary Function Channel: Press menu button PAGE_UP
pressMenuButton(PAGE_DOWN)	IMenuComponent	107			MENU_PAGE_DN	Momentary Function Channel: Press menu button PAGE_DOWN
pressMenuButton(DECK_A_B)	IMenuComponent	108			MENU_DECK_A_B	Momentary Function Channel: Press menu button DECK_A_B
pressMenuButton(TV_VCR)	IMenuComponent	109			MENU_TV_VCR	Momentary Function Channel: Press menu button TV_VCR

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton (RECORD_SPEED)	IMenuComponent	110			MENU_RECORD_SPEED	Momentary Function Channel: Press menu button RECORD_SPEED
pressMenuButton(PROGRAM)	IMenuComponent	111			MENU_PROGRAM	Momentary Function Channel: Press menu button PROGRAM
pressMenuButton(AB_REPEAT)	IMenuComponent	112			MENU_AB_REPEAT	Momentary Function Channel: Press menu button AB_REPEAT
pressMenuButton(HELP)	IMenuComponent	113			MENU_HELP	Momentary Function Channel: Press menu button HELP
pressMenuButton(TITLE)	IMenuComponent	114			MENU_TITLE	Momentary Function Channel: Press menu button TITLE
pressMenuButton(TOP_MENU)	IMenuComponent	115			MENU_TOP_MENU	Momentary Function Channel: Press menu button TOP_MENU
pressMenuButton(ZOOM)	IMenuComponent	116			MENU_ZOOM	Momentary Function Channel: Press menu button ZOOM
pressMenuButton(ANGLE)	IMenuComponent	117			MENU_ANGLE	Momentary Function Channel: Press menu button ANGLE
pressMenuButton(AUDIO)	IMenuComponent	118			MENU_AUDIO	Momentary Function Channel: Press menu button AUDIO
pressMenuButton (PREVIEW_INPUT)	IMenuComponent	129			MENU_PREVIEW_INPUT	Momentary Function Channel: Press menu button PREVIEW_INPUT
pressMenuButton(SEND_INPUT)	IMenuComponent	130			MENU_SEND_INPUT	Momentary Function Channel: Press menu button SEND_INPUT
pressMenuButton (SEND_GRAPHICS)	IMenuComponent	131			MENU_SEND_GRAPHICS	Momentary Function Channel: Press menu button SEND_GRAPHICS
pressMenuButton(FLASH)	IMenuComponent	203			MENU_FLASH	Momentary Function Channel: Press menu button FLASH
pressMenuButton(RESET)	IMenuComponent	215			MENU_RESET	Momentary Function Channel: Press menu button RESET
pressMenuButton (INSTANT_REPLAY)	IMenuComponent	218			MENU_INSTANT_REPLAY	Momentary Function Channel: Press menu button INSTANT_REPLAY
pressMenuButton(A)	IMenuComponent			ALPHA-A		Press menu button A
pressMenuButton(B)	IMenuComponent			ALPHA-B		Press menu button B
pressMenuButton(C)	IMenuComponent			ALPHA-C		Press menu button C
pressMenuButton(D)	IMenuComponent			ALPHA-D		Press menu button D
pressMenuButton(E)	IMenuComponent			ALPHA-E		Press menu button E
pressMenuButton(F)	IMenuComponent			ALPHA-F		Press menu button F
pressMenuButton(G)	IMenuComponent			ALPHA-G		Press menu button G
pressMenuButton(H)	IMenuComponent			ALPHA-H		Press menu button H

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
pressMenuButton(I)	IMenuComponent			ALPHA-I		Press menu button I
pressMenuButton(J)	IMenuComponent			ALPHA-J		Press menu button J
pressMenuButton(K)	IMenuComponent			ALPHA-K		Press menu button K
pressMenuButton(L)	IMenuComponent			ALPHA-L		Press menu button L
pressMenuButton(M)	IMenuComponent			ALPHA-M		Press menu button M
pressMenuButton(N)	IMenuComponent			ALPHA-N		Press menu button N
pressMenuButton(O)	IMenuComponent			ALPHA-O		Press menu button O
pressMenuButton(P)	IMenuComponent			ALPHA-P		Press menu button P
pressMenuButton(Q)	IMenuComponent			ALPHA-Q		Press menu button Q
pressMenuButton(R)	IMenuComponent			ALPHA-R		Press menu button R
pressMenuButton(S)	IMenuComponent			ALPHA-S		Press menu button S
pressMenuButton(T)	IMenuComponent			ALPHA-T		Press menu button T
pressMenuButton(U)	IMenuComponent			ALPHA-U		Press menu button U
pressMenuButton(V)	IMenuComponent			ALPHA-V		Press menu button V
pressMenuButton(W)	IMenuComponent			ALPHA-W		Press menu button W
pressMenuButton(X)	IMenuComponent			ALPHA-X		Press menu button X
pressMenuButton(Y)	IMenuComponent			ALPHA-Y		Press menu button Y
pressMenuButton(Z)	IMenuComponent			ALPHA-Z		Press menu button Z
selectMenuItem()	IMenuComponent	49			MENU_SELECT	Momentary Function Channel: Select current menu item
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cycleInputSelect()	ISourceSelectComponent			CYCLESELECT		Cycles to the next input select which has a different Group number than the current Group number, then responds with INPUTSELECT-<index>
getInputCount()	ISourceSelectComponent			?INPUTCOUNT		Query for the number of inputs
getInputProperties()	ISourceSelectComponent			?INPUTPROPERTIES		Query input properties for all inputs, responds with INPUTPROPERTIES-<index>,<inputGroup>, <signalType>,<deviceLabel>,<displayName>[; <index>,<inputGroup>,<signalType>,<deviceLabel>, <displayName>]*
getInputProperty(index)	ISourceSelectComponent			?INPUTPROPERTY-<index>		Query input properties for single input, responds with INPUTPROPERTY-<index>,<inputGroup>, <signalType>,<deviceLabel>,<displayName>, where <index> is a virtual input number between 1 and <count>, <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <signalType> is a TSE with type values for RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB, <deviceLabel> is the label on the device, and <displayName> is the text shown on the device display.
getInputSelect()	ISourceSelectComponent			?INPUTSELECT		Gets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setInputSelect(index)	ISourceSelectComponent			INPUTSELECT-<index>		Sets the current input, where <index> is a virtual input number between 1 and the value returned by ?INPUTCOUNT, responds with INPUTSELECT-<index>
processInputCountEvent(count)	ISourceSelectComponentListener			INPUTCOUNT-<count>		Response to ?INPUTCOUNT, where <count> is the integer number of inputs
processInputPropertiesEvent (String)	ISourceSelectComponentListener			INPUTPROPERTIES-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName> [;<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>]*		Returns comma-separated string containing <index>,<inputGroup>,<signalType>, <deviceLabel>,<displayName> for each input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), <deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputPropertyEvent (String)	ISourceSelectComponentListener			INPUTPROPERTY-<index>,<inputGroup>, <signalType>, <deviceLabel>, <displayName>		Returns comma-separated string containing <index>,<inputGroup>,<signalType>,<deviceLabel>, <displayName> for selected input. Where <inputGroup> is the integer value of the virtual input port on the device (mutually exclusive group), < deviceLabel> is the label on the device, <displayName> is the text shown on the device display and <signalType> is RGB, SVIDEO, COMPOSITE, COMPONENT, DVI, HDMI, SDI, VGA, AUDIO, LINE, or USB
processInputSelectEvent(index)	ISourceSelectComponentListener			INPUTSELECT-<index>		Returns integer index of currently selected input
getProjectorPreset()	IVideoProjectorComponent			?VPROJPRESET		Query for projector preset, responds with VPROJPRESET-<preset>
setProjectorPreset(preset)	IVideoProjectorComponent			VPROJPRESET-<preset>		Recall projector preset where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processProjectorPresetEvent(int)	IVideoProjectorComponentListener			VPROJPRESET-<int>		Projector preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255

Video Projector Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent(boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent (DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Video Projector State Charts

processLampPowerEvent		
State	**Channel 253**	**Channel 255**
OFF	OFF	OFF
COOL	ON	OFF
ON	OFF	ON
WARM	ON	ON

***Note**: Refer to "Appendix - Deprecated Functions" on page 349 for a listing of deprecated device functions associated with this device.*

Video Wall

- IModuleComponent
- IModuleComponentListener
- IPowerComponent
- IPowerComponentListener
- IVideoWallComponent
- IVideoWallComponentListener
- IWindowComponent
- IWindowComponentListener

Name: Video Wall						
Interface: IVideoWall						
Video Wall Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied file-name. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent(boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO

Video Wall Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDeviceOnLineEvent(boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cyclePower()	IPowerComponent	9			POWER	Momentary Function Channel: Cycle power when channel is activated
setPower(ON)	IPowerComponent	27			PWR_ON	Momentary Function Channel: Power is turned on when channel is activated
setPower(OFF)	IPowerComponent	28			PWR_OFF	Momentary Function Channel: Power is turned off when channel is activated
setPower(ps)	IPowerComponent	255			POWER_ON	Discrete Function Channel: Power is on while channel is active
setPowerSensor(nId,nIOChan)	IPowerComponent			IOLINK-<nId>,<channel>		Associate a Power Sensor with the device where <dps> is the DPS in string form, i.e. 17:1:0, and <channel> is the channel on the IO device to which the power sensor is connected.
processPowerEvent(ON)	IPowerComponentListener	255			POWER_FB	Feedback Channel: Power state changed, power is on while channel is on
getVideoWallConfiguration()	IVideoWallComponent			?VWALLCONFIG		Query for video wall configuration, responds with VWALLCONFIG-<config>
queryVideoWallConfigurationList()	IVideoWallComponent			?VWALLCONFIGLIST		Query for the list of available Video Wall Configurations, responds with multiple VWALLCONFIGLIST-<config> commands, one for each valid configuration
saveVideoWallConfiguration(sConfig)	IVideoWallComponent			VWALLCONFIGSAVE-<config>		Save video wall configuration where <config> is the configuration name
setVideoWallConfiguration(sConfig)	IVideoWallComponent			VWALLCONFIG-<config>		Recall video wall configuration where <config> is the configuration name
processVideoWallConfigurationEvent (String)	IVideoWallComponentListener			VWALLCONFIG-<config>		Video wall configuration changed, where <config> is the configuration name
processVideoWallConfigurationListEvent (String[])	IVideoWallComponentListener			VWALLCONFIGLIST-<config>		Response to Video Wall configuration list query, responds with multiple VWALLCONFIGLIST-<config> commands, one for each valid configuration
addWindowComponent (index,windowAddress)	IWindowComponent			WINDOWADD-<index>, <windowAddress>		Add a window at a given index, where <index> is 1 through x and <address> is a window address and x is the maximum supported window index (see specific module documentation)

Video Wall Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
adjustBrightness(1)	IWindowComponent	148			BRIGHT_UP	Momentary Function Channel: Brightness is incremented when channel is activated
adjustBrightness(-1)	IWindowComponent	149			BRIGHT_DN	Momentary Function Channel: Brightness is decremented when channel is activated
adjustColor(1)	IWindowComponent	150			COLOR_UP	Momentary Function Channel: Color is incremented when channel is activated
adjustColor(-1)	IWindowComponent	151			COLOR_DN	Momentary Function Channel: Color is decremented when channel is activated
adjustContrast(1)	IWindowComponent	152			CONTRAST_UP	Momentary Function Channel: Contrast is incremented when channel is activated
adjustContrast(-1)	IWindowComponent	153			CONTRAST_DN	Momentary Function Channel: Contrast is decremented when channel is activated
adjustSharpness(1)	IWindowComponent	154			SHARP_UP	Momentary Function Channel: Sharpness is incremented when channel is activated
adjustSharpness(-1)	IWindowComponent	155			SHARP_DN	Momentary Function Channel: Sharpness is decremented when channel is activated
adjustTint(1)	IWindowComponent	156			TINT_UP	Momentary Function Channel: Tint is incremented when channel is activated
adjustTint(-1)	IWindowComponent	157			TINT_DN	Momentary Function Channel: Tint is decremented when channel is activated
cycleFreeze()	IWindowComponent	213			PIC_FREEZE	Momentary Function Channel: Cycle freeze when channel is activated
cyclePictureMute()	IWindowComponent	210			PIC_MUTE	Momentary Function Channel: Cycle picture mute when channel is activated
getImagePosition()	IWindowComponent			?IMAGE		Query for the image position. Response with IMAGE-<source>,<x>,<y>,<height>,<width> where <source> is 1 to the maximum source number (see specific module documentation)
getImageSize()	IWindowComponent			?IMAGE		Query for the image size. Response with IMAGE-<source>,<x>,<y>,<height>,<width> where <source> is 1 to the maximum source number (see specific module documentation)
getImageSource()	IWindowComponent			?IMAGE		Query for the image source. Response with IMAGE-<source>,<x>,<y>,<height>,<width> where <source> is 1 to the maximum source number (see specific module documentation)
getWindowComponentAddress(index)	IWindowComponent			?WINDOWADDR-<index>		Query for the address of the window at index <index>, responds with WINDOWADDR-<index>,<address>
getWindowComponentIndex (windowAddress)	IWindowComponent			?WINDOWIDX-<address>		Query for the index of the window with address <address>, responds with WINDOWADDR-<index>,<address>

Video Wall Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getWindowPosition()	IWindowComponent			?WINDOW		Query for the window position. Response with WINDOW-<x>,<y>,<height>,<width>
getWindowSize()	IWindowComponent			?WINDOW		Query for the window size. Response with WINDOW-<x>,<y>,<height>,<width>
getZOrder()	IWindowComponent			?WINDOWZORDER		Get the Z-order. Responds with WINDOWZORDER-<value> where <value> is 1 to the maximum z-order of the device, 1 is the top-most z-order (see specific module documentation)
pan(UP)	IWindowComponent	132			PAN_UP	Momentary Function Channel: Image is moved up one step within the window when channel is activated
pan(DOWN)	IWindowComponent	133			PAN_DN	Momentary Function Channel: Image is moved down one step within the window when channel is activated
pan(LEFT)	IWindowComponent	134			PAN_LT	Momentary Function Channel: Image is moved left one step within the window when channel is activated
pan(RIGHT)	IWindowComponent	135			PAN_RT	Momentary Function Channel: Image is moved right one step within the window when channel is activated
removeWindowComponent (windowAddress)	IWindowComponent			WINDOWREMOVEADDR-<windowAddress>		Remove the window with address <address>, where <address> is a window address
removeWindowComponent(index)	IWindowComponent			WINDOWREMOVEIDX-<index>		Remove the window at index <index>, where <index> is 1 through x and x is the maximum supported window index (see specific module documentation)
setBrightness(level)	IWindowComponent		10		BRIGHT_LVL	Set brightness level, range is 0-255
setColor(level)	IWindowComponent		11		COLOR_LVL	Set color level, range is 0-255
setContrast(level)	IWindowComponent		12		CONTRAST_LVL	Set contrast level, range is 0-255
setFreezeOn(state)	IWindowComponent	214			PIC_FREEZE_ON	Discrete Function Channel: Freeze is on while channel is active
setImage(source,pos,size)	IWindowComponent			IMAGE-<source>,<x>,<y>,<height>,<width>		Set the image source, position and size relative to the window where <source> is 1 to the maximum source number (see specific module documentation) and <x> and <y> are the window coordinates for the upper-left hand corner of the image and <height> and <width> are the size of the image to be displayed in the window. Used to change the position of the image visible in the window.
setImageSource(int)	IWindowComponent			IMAGESOURCE-<source>		Set the image source for the window where <source> is 1 to the maximum source number (see specific module documentation)
setPictureMuteOn(state)	IWindowComponent	211			PIC_MUTE_ON	Discrete Function Channel: Picture Mute is on while channel is active
setSharpness(level)	IWindowComponent		13		SHARP_LVL	Set sharpness level, range is 0-255
setTint(level)	IWindowComponent		14		TINT_LVL	Set tint level, range is 0-255

Video Wall Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
setWindow(pos,size)	IWindowComponent			WINDOW-<x>,<y>,<height>,<width>		Set the window position and size relative to the display where <x> and <y> are the display coordinates for the upper-left hand corner of the window and <height> and <width> are the size of the window to be displayed in the display. Used to change the position of the window visible in the display.
setZOrder(position)	IWindowComponent			WINDOWZORDER-<position>		Shuffle the window z-order relative to the other windows where <position> is FRONT, BACK, FORWARD (up one level), BACKWARD (down one level)
setZOrder(value)	IWindowComponent			WINDOWZORDER-<value>		Set the Z-order where <value> is 1 to the maximum z-order of the device, 1 is the top-most z-order (see specific module documentation)
zoomIn()	IWindowComponent	159			ZOOM_IN	Momentary Function Channel: Image size is enlarged within the window (zoom in/tele) when channel is activated
zoomOut()	IWindowComponent	158			ZOOM_OUT	Momentary Function Channel: Image size is reduced within the window (zoom out/window) when channel is activated
processBrightnessEvent(int)	IWindowComponentListener		10		BRIGHT_LVL	Brightness changed, range is 0-255
processColorEvent(int)	IWindowComponentListener		11		COLOR_LVL	Color changed, range is 0-255
processContrastEvent(int)	IWindowComponentListener		12		CONTRAST_LVL	Contrast changed, range is 0-255
processFreezeEvent(boolean)	IWindowComponentListener	214			PIC_FREEZE_FB	Feedback Channel: Freeze is on if channel is on
processImageEvent(Position,Size)	IWindowComponentListener			IMAGE-<source>,<x>,<y>,<height>,<width>		Image position and/or size changed where <source> is 1 to the maximum source number (see specific module documentation)
processImageSourceEvent(int)	IWindowComponentListener			IMAGESOURCE-<source>		Image source for the window changed where <source> is 1 to the maximum source number (see specific module documentation)
processPictureMuteEvent(boolean)	IWindowComponentListener	211			PIC_MUTE_FB	Feedback Channel: Picture is muted if channel is on
processSharpnessEvent(int)	IWindowComponentListener		13		SHARP_LVL	Sharpness changed, range is 0-255
processTintEvent(int)	IWindowComponentListener		14		TINT_LVL	Tint changed, range is 0-255
processWindowEvent(Position,Size)	IWindowComponentListener			WINDOW-<x>,<y>,<height>,<width>		Window position and/or size changed.
processZOrderEvent(int)	IWindowComponentListener			WINDOWZORDER-<value>		Z-order changed where <value> is 1 to the maximum z-order of the device, 1 is the top-most z-order (see specific module documentation)

Video Wall State Charts

processPowerEvent	
State	**Channel 255**
OFF	OFF
ON	ON

Volume Controller

- IModuleComponent
- IModuleComponentListener
- IVolumeComponent
- IVolumeControllerComponent
- IVolumeControllerComponentListener
- IVolumeComponentListener

Name: Volume Controller						
Interface: IVolumeController						
Volume Controller Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied filename. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on

Volume Controller Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
cycleVolumeMute()	IVolumeComponent	26			VOL_MUTE	Momentary Function Channel: Cycle volume mute when channel is activated
cycleVolumePreset()	IVolumeComponent	138			VOL_PRESET	Momentary Function Channel: Cycle camera preset when channel is activated
getVolumePreset()	IVolumeComponent			?VOLPRESET		Query for volume preset, responds with VOLPRESET-<preset>
saveVolumePreset(preset)	IVolumeComponent			VOLPRESETSAVE-<preset>		Save Volume Preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolume(level)	IVolumeComponent		1		VOL_LVL	Set volume, range is 0-255
setVolumeMuteOn(state)	IVolumeComponent	199			VOL_MUTE_ON	Discrete Function Channel: Volume mute is on while channel is active
setVolumePreset(preset)	IVolumeComponent			VOLPRESET-<preset>		Recall volume preset where <preset> is 1 to x and x is the maximum supported preset (see specific module documentation)
setVolumeRamp(UP)	IVolumeComponent	24			VOL_UP	Ramping Channel: Volume is ramped up while channel is active
setVolumeRamp(DOWN)	IVolumeComponent	25			VOL_DN	Ramping Channel: Volume is ramped down while channel is active
processVolumeEvent(int)	IVolumeComponentListener		1		VOL_LVL	Volume changed, range is 0-255
processVolumeMuteEvent (boolean)	IVolumeComponentListener	199			VOL_MUTE_FB	Feedback Channel: Volume is muted if channel is on
processVolumePresetEvent(int)	IVolumeComponentListener			VOLPRESET-<preset>		Volume preset changed, where <preset> is 1-x and x is the maximum supported preset (see specific module documentation)
processVolumeRampEvent(UP)	IVolumeComponentListener	24			VOL_UP_FB	Feedback Channel: Volume is ramping up while channel is on
processVolumeRampEvent (DOWN)	IVolumeComponentListener	25			VOL_DN_FB	Feedback Channel: Volume is ramping down while channel is on

Weather

- IModuleComponent
- IModuleComponentListener
- IWeatherComponent
- IWeatherComponentListener

Name: Weather						
Interface: IWeather						
Weather Functions						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getDebugState()	IModuleComponent			?DEBUG		Query the debug level, responds with DEBUG-<state> where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
getFWVersion()	IModuleComponent			?FWVERSION		Query for the device firmware version, responds with FWVERSION-<version>
getProperty(key)	IModuleComponent			?PROPERTY-<key>		Query for the value of property <key>, respond with PROPERTY-<key>,<value>
getVersion()	IModuleComponent			?VERSION		Query for the module version, responds with VERSION-<version>
loadProperties(fileName)	IModuleComponent			LOADPROPERTY-<file>		Load properties from the file with the supplied file-name. Used to load meta-data properties for modules including Generic IR modules.
passThru(buffer)	IModuleComponent			PASSTHRU-<buffer>		Send a message directly to the device
reinitialize()	IModuleComponent			REINIT		Reinitialize communication with the device
setDebugState(state)	IModuleComponent			DEBUG-<state>		Set the debug state where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
setDeviceDateTime(date)	IModuleComponent			CLOCK-<mm/dd/yyyy> <hh:mm:ss>		Set the device date/time.
setPassbackOn(boolean)	IModuleComponent			PASSBACK-<state>		Set the passback state where <state> is 1 or 0. When passback is on, all response from the device will be passed back to the NetLinx program as a string from the virtual device
setProperty(key,value)	IModuleComponent			PROPERTY-<key>, <value>		Set the value of property <key> to <value>
processDataInitializedEvent (boolean)	IModuleComponentListener	252			DATA_INITIALIZED	Feedback Channel: Module data is synchronized with device while channel is on
processDebugEvent(int)	IModuleComponentListener			DEBUG-<state>		Debug state changed where <state> is 1-4 for ERROR, WARNING, DEBUG, INFO
processDeviceOnLineEvent (boolean)	IModuleComponentListener	251			DEVICE_COMMUNICATING	Feedback Channel: Communication is established with device while channel is on

Weather Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processPassbackEvent(byte[])	IModuleComponentListener					When passback is on, each string received form the device is sent to the NetLinx program as a string. Use a DATA_EVENT event with a STRING handler to capture the data from the device.
getChanceOfPrecipitation()	IWeatherComponent			?FORECASTCOP		Query for forecast chance of precipitation changes, responds with FORECASTCOP-<day 1>,<day 2>,etc... Value list will contain one forecast chance of precipitation for each forecast day. The first day is always today. Values are in percent.
getCurrentCondition()	IWeatherComponent			?WEATHERCONDITION		Query for current conditions, responds with WEATHERCONDITION-<condition>, <condition> will be BLIZZARD, BLOWINGSNOW, CLEAR, CLOUDY, DRIZZLE, DUST, FAIR, FOG, FREEZINGDRIZZLE, FREEZINGRAIN, HAZE, HUMID, ICE, MOSTLYCLOUDY, MOSTLYSUNNY, PARTLYCLOUDY, RAIN, RAINSHOWERS, RAINSNOWMIX, SLEET, SMOKE, SNOW, SNOWFLURRIES, SNOWSHOWERS, SUNNY, THUNDERSTORMS, UNKNOWN, VERYCOLD, WINDY
getForecastHighTemperature()	IWeatherComponent			?FORECASTHIGH		Query for forecast high temperatures changes, responds with FORECASTHIGH-<day 1>,<day 2>, etc... Value list will contain one forecast high temperature for each forecast day. The first day is always today. Values are in degrees C or F depending on weather scale
getForecastLowTemperature()	IWeatherComponent			?FORECASTLOW		Query for forecast low temperatures changes, responds with FORECASTLOW-<day 1>,<day 2>, etc... Value list will contain one forecast high temperature for each forecast day. The first day is always today. Values are in degrees C or F depending on weather scale
getRainfall(dur)	IWeatherComponent			?WEATHERRAIN-<duration>		Query for Rain fall, responds with WEATHERRAIN-<duration>,<value> where <duration> is DAY, WEEK, MONTH, YEAR, YTD and <value> is in inches (Imperial Scale) or cm (Metric Scale).
getWeatherAlert()	IWeatherComponent			?WEATHERALERT		Query for Weather alert, responds with WEATHERALERT-<alert> where <alert> is a string containing the weather alert

Weather Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
getWeatherConditions()	IWeatherComponent			?FORECAST CONDITION		Query for forecast conditions, responds with FORECASTCONDITION-<day 1>,<day 2>, etc. Value list will contain one forecast condition for each forecast day. The first day is always today. Values will be BLIZZARD, BLOWINGSNOW, CLEAR, CLOUDY, DRIZZLE, DUST, FAIR, FOG, FREEZINGDRIZZLE, FREEZINGRAIN, HAZE, HUMID, ICE, MOSTLYCLOUDY, MOSTLYSUNNY, PARTLY-CLOUDY, RAIN, RAINSHOWERS, RAINSNOWMIX, SLEET, SMOKE, SNOW, SNOWFLURRIES, SNOWSHOWERS, SUNNY, THUNDERSTORMS, UNKNOWN, VERYCOLD, WINDY
getWeatherScale()	IWeatherComponent			?WEATHERSCALE		Query for the temperature scale, responds with WEATHERSCALE-<scale> where <scale> is IMPERIAL or METRIC
getWindInfo()	IWeatherComponent			?WEATHERWIND		Query for the wind speed/direction, responds with WEATHERWIND-<speed>,direction> where <speed> will be in mph or kph, depending on scale and <direction> will be N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW
queryWeatherAll()	IWeatherComponent	208			WEATHER_FORCE_READING	Momentary Function Channel: Causes the weather station to update its readings when the channel is activated
setWeatherScale(us)	IWeatherComponent			WEATHERSCALE-<scale>		Set the weather scale, where <scale> is IMPERIAL or METRIC
processBarometricPressureEvent(float)	IWeatherComponentListener		48		WEATHER_BAR_LVL	Barometric pressure changed, value is in inches Hg (Imperial Scale) or mm Hg/torr (Metric Scale)
processBarometricTrendEvent(RISING)	IWeatherComponentListener	232			WEATHER_BAR_RISING	Feedback Channel: Barometric pressure trend change (see chart below)
processBarometricTrendEvent (FALLING)	IWeatherComponentListener	233			WEATHER_BAR_FALLING	Feedback Channel: Barometric pressure trend change (see chart below)
processChanceOfPrecipitationEvent (int[])	IWeatherComponentListener			FORECASTCOP-<day1>[,<day2>,...]		Weather forecast chance of precipitation changes, value list will contain one forecast chance of precipitation for each forecast day. The first day is always today. Values are in percent.
processCurrentConditionEvent (WeatherConditions)	IWeatherComponentListener			WEATHERCONDITION-<condition>		Weather condition changed, value will be BLIZZARD, BLOWINGSNOW, CLEAR, CLOUDY, DRIZZLE, DUST, FAIR, FOG, FREEZINGDRIZZLE, FREEZINGRAIN, HAZE, HUMID, ICE, MOSTLYCLOUDY, MOSTLYSUNNY, PARTLY-CLOUDY, RAIN, RAINSHOWERS, RAINSNOWMIX, SLEET, SMOKE, SNOW, SNOWFLURRIES, SNOWSHOWERS, SUNNY, THUNDERSTORMS, UNKNOWN, VERYCOLD, WINDY

Weather Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processDewpointEvent(float)	IWeatherComponentListener		47		WEATHER_DEWPOINT_LVL	Dewpoint changed, value is in degrees C or F depending on weather scale
processForecastHighTemperatureEvent (int[])	IWeatherComponentListener			FORECASTHIGH-<day1>[,<day2>,...]		Weather forecast high temperatures changes, value list will contain one forecast high temperature for each forecast day. The first day is always today. Values are in degrees C or F depending on weather scale
processForecastLowTemperatureEvent (int[])	IWeatherComponentListener			FORECASTLOW-<day1>[,<day2>,...]		Weather forecast low temperatures changes, value list will contain one forecast low temperature for each forecast day. The first day is always today. Values are in degrees C or F depending on weather scale
processHeatIndexEvent(float)	IWeatherComponentListener		46		WEATHER_HEAT_INDEX_LVL	Heat index temperature changed, value is in degrees C or F depending on weather scale
processHighTemperatureEvent(float)	IWeatherComponentListener		43		WEATHER_HI_TEMP_LVL	High temperature since midnight changed, value is in degrees C or F depending on weather scale
processIndoorHumidityEvent(int)	IWeatherComponentListener		35		INDOOR_HUMID_LVL	Indoor humidity changed, value is in percent
processIndoorTemperatureEvent(float)	IWeatherComponentListener		33		INDOOR_TEMP_LVL	Indoor temperature changed, value is in degrees C or F depending on weather scale
processIsFreezingEvent(boolean)	IWeatherComponentListener	231			WEATHER_FREEZING	Feedback Channel: Weather condition is freezing if channel is on. Weather condition is freezing when outdoor temperature is at or below 32 degrees F, 0 degrees C.
processIsRainingEvent(boolean)	IWeatherComponentListener	230			WEATHER_RAINING	Feedback Channel: Weather condition is raining if channel is on. Weather condition is raining when the current weather condition indicates raining.
processLowTemperatureEvent(float)	IWeatherComponentListener		44		WEATHER_LO_TEMP_LVL	Low temperature since midnight changed, value is in degrees C or F depending on weather scale
processOutdoorHumidityEvent(int)	IWeatherComponentListener		36		OUTDOOR_HUMID_LVL	Outdoor humidity changed, value is in percent
processOutdoorTemperatureEvent (float)	IWeatherComponentListener		34		OUTDOOR_TEMP_LVL	Outdoor temperature changed, value is in degrees C or F depending on weather scale
processRainfallEvent(float,Duration)	IWeatherComponentListener			WEATHERRAIN-<duration>,<value>		Rain fall changed, where <duration> is DAY, WEEK, MONTH, YEAR, YTD and <value> is in inches (Imperial Scale) or cm (Metric Scale).
processWeatherAlertEvent(String)	IWeatherComponentListener			WEATHERALERT-<alert>		Weather alert, where <alert> is a string containing the weather alert

Weather Functions (Cont.)						
Name	**Component**	**Channel**	**Level**	**Command**	**Constant**	**Notes**
processWeatherConditionsEvent (WeatherConditions[])	IWeatherComponentListener			FORECASTCONDITION-<day1>[,<day2>,...]		Weather forecast conditions changed, value list will contain one forecast condition for each forecast day. The first day is always today. Values will be BLIZZARD, BLOWINGSNOW, CLEAR, CLOUDY, DRIZZLE, DUST, FAIR, FOG, FREEZINGDRIZZLE, FREEZINGRAIN, HAZE, HUMID, ICE, MOSTLY-CLOUDY, MOSTLYSUNNY, PARTLYCLOUDY, RAIN, RAINSHOWERS, RAINSNOWMIX, SLEET, SMOKE, SNOW, SNOWFLURRIES, SNOWSHOWERS, SUNNY, THUNDERSTORMS, UNKNOWN, VERYCOLD, WINDY
processWeatherScaleEvent(UnitScale)	IWeatherComponentListener			WEATHERSCALE-<scale>		Weather scale changed, <scale> is IMPERIAL or METRIC
processWindchillEvent(float)	IWeatherComponentListener		45		WEATHER_WIND_CHILL_LVL	Windchill temperature changed, value is in degrees C or F depending on weather scale
processWindInfoEvent (WindInfo)	IWeatherComponentListener			WEATHERWIND-<speed>,direction>		Wind speed/direction had changed. Wind speed will be in mph or kph, depending on scale. Direction will be N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW

Weather State Charts

processBarometricTrendEvent		
State	**Channel 232**	**Channel 233**
STEADY	OFF	OFF
RISING	ON	OFF
FALLING	OFF	ON

Appendix - Deprecated Functions

The Device Functions listed in this section are considered to be deprecated functions. Refer to the *Notes* column for information on the preferred method for each deprecated function.

Deprecated Functions					
Name	**Component**	**Command**	**Constant**	**Notes**	**Devices**
cycleInputSource()	ISourceSelectComponent		SOURCE_CYCLE	Momentary Function Channel: Cycle input source when channel is activated **deprecated Input Source in favor of Input Select** **Channel = 196**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
getInputGroupSelect()	ISourceSelectComponent	?INPUTGROUPSELECT		Query for currently selected input group. **deprecated Input Group Select in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
getInputSource()	ISourceSelectComponent	?INPUT		Query current input, responds with INPUT-<sourceSelect>,<inputNumber> where <sourceSelect> is RGB,SVIDEO,COMPOSITE, COMPONENT,DVI,HDMI,SDI,VGA,AUDIO,AUXILIARY, CABLE,CAMERA,CD,DVD,FRONT,HDTV,LASERDISC, LINE,MEDIAPLAYER,MINIDISC,PHONO,SATELLITE, TAPE,TUNER,TV,VCR,VIDEO <inputNumber> is the instance number of the source select. **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector

Deprecated Functions (Cont.)					
Name	**Component**	**Command**	**Constant**	**Notes**	**Devices**
setInputGroupSelect (group)	ISourceSelectComponent	INPUTGROUPSELECT-<group>		Set input group selection. **deprecated Input Group Select in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (TV,1)	ISourceSelectComponent		SOURCE_TV1	Momentary Function Channel: Selects input TV,1 when channel is activated. **Channel = 30** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (VIDEO,1)	ISourceSelectComponent		SOURCE_VIDEO1	Momentary Function Channel: Selects input VIDEO,1 when channel is activated. **Channel = 31** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector

Deprecated Functions (Cont.)					
Name	**Component**	**Command**	**Constant**	**Notes**	**Devices**
setInputSource (VIDEO,2)	ISourceSelectComponent		SOURCE_VIDEO2	Momentary Function Channel: Selects input VIDEO,2 when channel is activated. **Channel = 32** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (VIDEO,3)	ISourceSelectComponent		SOURCE_VIDEO3	Momentary Function Channel: Selects input VIDEO,3 when channel is activated. **Channel = 33** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (TAPE,1)	ISourceSelectComponent		SOURCE_TAPE1	Momentary Function Channel: Selects input TAPE,1 when channel is activated. **Channel = 34** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector

Deprecated Functions (Cont.)					
Name	**Component**	**Command**	**Constant**	**Notes**	**Devices**
setInputSource (TAPE,2)	ISourceSelectComponent		SOURCE_TAPE2	Momentary Function Channel: Selects input TAPE,2 when channel is activated. **Channel = 35** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (CD,1)	ISourceSelectComponent		SOURCE_CD1	Momentary Function Channel: Selects input CD,1 when channel is activated. **Channel = 36** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (TUNER,1)	ISourceSelectComponent		SOURCE_TUNER1	Momentary Function Channel: Selects input TUNER,1 when channel is activated. **Channel = 37** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector

Deprecated Functions (Cont.)					
Name	**Component**	**Command**	**Constant**	**Notes**	**Devices**
setInputSource (PHONO,1)	ISourceSelectComponent		SOURCE_PHONO1	Momentary Function Channel: Selects input PHONO,1 when channel is activated. **Channel = 38** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (AUXILIARY,1)	ISourceSelectComponent		SOURCE_AUX1	Momentary Function Channel: Selects input AUXILIARY,1 when channel is activated. **Channel = 39** **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
setInputSource (sourceSelect,inputNumber)	ISourceSelectComponent	INPUT-<sourceSelect>, <inputNumber>		Set the current input, where <sourceSelect> is RGB,SVIDEO,COMPOSITE, COMPONENT,DVI,HDMI,SDI,VGA,AUDIO,AUXILIARY, CABLE,CAMERA,CD,DVD,FRONT,HDTV,LASERDISC,LINE, MEDIAPLAYER,MINIDISC,PHONO,SATELLITE, TAPE,TUNER,TV,VCR, VIDEO <inputNumber> is the instance number of the source select. **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector

Deprecated Functions (Cont.)					
Name	**Component**	**Command**	**Constant**	**Notes**	**Devices**
processInputGroupSelectEvent (group)	ISourceSelectComponentListener	INPUTGROUPSELECT-<inputGroup>		Returns integer index of currently selected input group. **deprecated Input Group Select in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector
processInputSourceEvent (InputSourceSelectInfo)	ISourceSelectComponentListener	INPUT-<sourceSelect>, <inputNumber>		Current input has changed, where <sourceSelect> is RGB,SVIDEO,COMPOSITE, COMPONENT,DVI,HDMI,SDI,VGA,AUDIO,AUXILIARY, CABLE,CAMERA,CD, COMPUTER, DVD,FRONT,HDTV, LASERDISC,LINE,MEDIAPLAYER,MINIDISC,PHONO, SATELLITE,TAPE,TUNER,TV,VCR,VIDEO <inputNumber> is the instance number of the source select. **deprecated Input Source in favor of Input Select**	• Digital Video Recorder • Document Camera • Monitor • Pre Amp Surround Sound Processor • Receiver • TV • VCR • Video Conferencer • Video Processor • Video Projector

Increase Your Revenue through education + knowledge

In the ever-changing AV industry, continual education is key to success. AMX University is dedicated to ensuring that you have the opportunity to gather the information and experience you need to deliver strong AMX solutions. Plus, AMX courses also help you earn CEDIA, NSCA, InfoComm, and AMX continuing education units (CEUs).

Visit AMX University online for 24/7/365 access to:

- *Schedules and registration for any AMX University course*
- *Travel and hotel information*
- *Your individual certification requirements and progress*

1/11 ©2011 AMX. All rights reserved. AMX and the AMX logo are registered trademarks of AMX. AMX reserves the right to alter specifications without notice at any time.

3000 RESEARCH DRIVE, RICHARDSON, TX 75082 USA • 800.222.0193 • 469.624.8000 • 469-624-7153 fax • 800.932.6993 technical support • www.amx.com

93-0070-60 REV B