PROGRAMMERS GUIDE

NMX-MM-1000
ENZO™ MEETING PRESENTATION SYSTEM

AV FOR AN IT WORLD

AMX Limited Warranty and Disclaimer

This Limited Warranty and Disclaimer extends only to products purchased directly from AMX or an AMX Authorized Partner which include AMX Dealers, Distributors, VIP's or other AMX authorized entity.

AMX warrants its products to be free of defects in material and workmanship under normal use for three (3) years from the date of purchase, with the following exceptions:

- Electroluminescent and LCD Control Panels are warranted for three (3) years, except for the display and touch overlay components are warranted for a period of one (1) year.
- Disk drive mechanisms, pan/tilt heads, power supplies, and MX Series products are warranted for a period of one (1) year.
- AMX lighting products are guaranteed to switch on and off any load that is properly connected to our lighting products, as long as the AMX lighting products are under warranty. AMX also guarantees the control of dimmable loads that are properly connected to our lighting products. The dimming performance or quality there of is not guaranteed, impart due to the random combinations of dimmers, lamps and ballasts or transformers.
- AMX software is warranted for a period of ninety (90) days.
- Batteries and incandescent lamps are not covered under the warranty.
- AMX AutoPatch Epica, Modula, Modula Series4, Modula CatPro Series and 8Y-3000 product models will be free of defects in materials and manufacture at the time of sale and will remain in good working order for a period of three (3) years following the date of the original sales invoice from AMX. The three-year warranty period will be extended to the life of the product (Limited Lifetime Warranty) if the warranty card is filled out by the dealer and/or end user and returned to AMX so that AMX receives it within thirty (30) days of the installation of equipment but no later than six (6) months from original AMX sales invoice date. The life of the product extends until five (5) years after AMX ceases manufacturing the product model. The Limited Lifetime Warranty applies to products in their original installation only. If a product is moved to a different installation, the Limited Lifetime Warranty will no longer apply, and the product warranty will instead be the three (3) year Limited Warranty.

All products returned to AMX require a Return Material Authorization (RMA) number. The RMA number is obtained from the AMX RMA Department. The RMA number must be clearly marked on the outside of each box. The RMA is valid for a 30-day period. After the 30-day period the RMA will be canceled. Any shipments received not consistent with the RMA, or after the RMA is canceled, will be refused. AMX is not responsible for products returned without a valid RMA number.

AMX is not liable for any damages caused by its products or for the failure of its products to perform. This includes any lost profits, lost savings, incidental damages, or consequential damages. AMX is not liable for any claim made by a third party or by an AMX Authorized Partner for a third party.

This Limited Warranty does not apply to (a) any AMX product that has been modified, altered or repaired by an unauthorized agent or improperly transported, stored, installed, used, or maintained; (b) damage caused by acts of nature, including flood, erosion, or earthquake; (c) damage caused by a sustained low or high voltage situation or by a low or high voltage disturbance, including brownouts, sags, spikes, or power outages; or (d) damage caused by war, vandalism, theft, depletion, or obsolescence.

This limitation of liability applies whether damages are sought, or a claim is made, under this warranty or as a tort claim (including negligence and strict product liability), a contract claim, or any other claim. This limitation of liability cannot be waived or amended by any person. This limitation of liability will be effective even if AMX or an authorized representative of AMX has been advised of the possibility of any such damages. This limitation of liability, however, will not apply to claims for personal injury.

Some states do not allow a limitation of how long an implied warranty last. Some states do not allow the limitation or exclusion of incidental or consequential damages for consumer products. In such states, the limitation or exclusion of the Limited Warranty may not apply. This Limited Warranty gives the owner specific legal rights. The owner may also have other rights that vary from state to state. The owner is advised to consult applicable state laws for full determination of rights.

EXCEPT AS EXPRESSLY SET FORTH IN THIS WARRANTY, AMX MAKES NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. AMX EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED IN THIS LIMITED WARRANTY. ANY IMPLIED WARRANTIES THAT MAY BE IMPOSED BY LAW ARE LIMITED TO THE TERMS OF THIS LIMITED WARRANTY. EXCEPT AS OTHERWISE LIMITED BY APPLICABLE LAW, AMX RESERVES THE RIGHT TO MODIFY OR DISCONTINUE DESIGNS, SPECIFICATIONS, WARRANTIES, PRICES, AND POLICIES WITHOUT NOTICE.

IMPORTANT SAFETY INSTRUCTIONS!

1) READ these instructions.

2) KEEP these instructions.

3) HEED all warnings.

4) FOLLOW all instructions.

5) DO NOT use this apparatus near water.

6) CLEAN ONLY with dry cloth.

7) DO NOT block any ventilation openings. Install in accordance with the manufacturer's instructions.

8) DO NOT install near any heat sources such as radiators, heat registers, stoves, or other apparatus (including amplifiers) that produce heat.

9) DO NOT defeat the safety purpose of the polarized or grounding type plug. A polarized plug has two blades with one wider than the other. A grounding type plug has two blades and a third grounding prong. The wider blade or the third prong are provided for your safety. If the provided plug does not fit into your outlet, consult an electrician for replacement of the obsolete outlet.

10) PROTECT the power cord from being walked on or pinched, particularly at plugs, convenience receptacles, and the point where they exit from the apparatus.

11) ONLY USE attachments/accessories specified by the manufacturer.

12) USE ONLY with a cart, stand, tripod, bracket, or table specified by the manufacturer, or sold with the apparatus. When a cart is used, use caution when moving the cart/apparatus combination to avoid injury from tip-over.

13) UNPLUG this apparatus during lightning storms or when unused for long periods of time.

14) REFER all servicing to qualified service personnel. Servicing is required when the apparatus has been damaged in any way, such as power-supply cord or plug is damaged, liquid has been spilled or objects have fallen into the apparatus, the apparatus has been exposed to rain or moisture, does not operate normally, or has been dropped.

15) DO NOT expose this apparatus to dripping or splashing and ensure that no objects filled with liquids, such as vases, are placed on the apparatus.

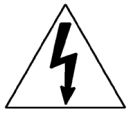
16) To completely disconnect this apparatus from the AC Mains, disconnect the power supply cord plug from the AC receptacle.

17) Where the mains plug or an appliance coupler is used as the disconnect device, the disconnect device shall remain readily operable.

18) DO NOT overload wall outlets or extension cords beyond their rated capacity as this can cause electric shock or fire.

(19) Place the equipment near a main power supply outlet and make sure that you can easily access the power breaker switch.

The exclamation point, within an equilateral triangle, is intended to alert the user to the presence of important operating and maintenance (servicing) instructions in the literature accompanying the product.



The lightning flash with arrowhead symbol within an equilateral triangle is intended to alert the user to the presence of uninsulated "dangerous voltage" within the product's enclosure that may be of sufficient magnitude to constitute a risk of electrical shock to persons.

ESD Warning: The icon to the left indicates text regarding potential danger associated with the discharge of static electricity from an outside source (such as human hands) into an integrated circuit, often resulting in damage to the circuit.

WARNING: To reduce the risk of fire or electrical shock, do not expose this apparatus to rain or moisture.

WARNING: No naked flame sources - such as candles - should be placed on the product.

WARNING: Equipment shall be connected to a MAINS socket outlet with a protective earthing connection.

WARNING: This product is intended to be operated ONLY from the voltages listed on the back panel or the recommended, or included, power supply of the product. Operation from other voltages other than those indicated may cause irreversible damage to the product and void the products warranty. The use of AC Plug Adapters is cautioned because it can allow the product to be plugged into voltages in which the product was not designed to operate. If the product is equipped with a detachable power cord, use only the type provided with your product or by your local distributor and/or retailer. If you are unsure of the correct operational voltage, please contact your local distributor and/or retailer.

ESD WARNING

To avoid ESD (Electrostatic Discharge) damage to sensitive components, make sure you are properly grounded before touching any internal materials.
When working with any equipment manufactured with electronic devices, proper ESD grounding procedures must be followed to make sure people, products, and tools are as free of static charges as possible. Grounding straps, conductive smocks, and conductive work mats are specifically designed for this purpose.
Anyone performing field maintenance on AMX equipment should use an appropriate ESD field service kit complete with at least a dissipative
work mat with a ground cord and a UL listed adjustable wrist strap with another ground cord

WARNING: Do Not Open! Risk of Electrical Shock. Voltages in this equipment are hazardous to life. No user-serviceable parts inside. Refer all servicing to qualified service personnel. Place the equipment near a main power supply outlet and make sure that you can easily access the power breaker switch.

FCC AND CANADA EMC COMPLIANCE INFORMATION:

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- •Reorient or relocate the receiving antenna.
- •Increase the separation between the equipment and receiver.
- •Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- •Consult the dealer or an experienced radio/TV technician for help.

Approved under the verification provision of FCC Part 15 as a Class B Digital Device.

Caution: Changes or modifications not expressly approved by the manufacturer could void the user's authority to operate this device.

CAN ICES-3 (B)/NMB-3(B)

EU COMPLIANCE INFORMATION:

Eligible to bear the CE mark; Conforms to European Union Low Voltage Directive 2006/95/EC; European Union EMC Directive 2004/108/EC; European Union Restriction of Hazardous Substances Recast (RoHS2) Directive 2011/65/EU.; European Union WEEE (recast) Directive 2012/19/EU; European Union Registration, Evaluation, Authorization and Restriction of Chemicals (REACH) Directive 2006/121/EC

You may obtain a free copy of the Declaration of Conformity by visiting http://www.amx.com/techcenter/certifications.asp

WEEE NOTICE

This appliance is labeled in accordance with European Directive 2002/96/EC concerning waste of electrical and electronic equipment (WEEE). This label indicates that this product should not be disposed of with household waste. It should be deposited at an appropriate facility to enable recovery and recycling.

Table of Contents

AMX Shell Commands

Overview

This document provides a an overview of the specific secured Shell commands and NetLinx commands used on the NMX-MM-1000 Enzo Meeting Presentation System (**FG3211-01**) platform for integration with a NetLinx system.

AMX Shell is a Java implementation of a command shell that can be implemented to expose remote access and management to the Enzo device. To access AMX Shell, the Enzo device must have Secure Shell (SSH) enabled. Refer to *Diagnostics* in the *Enzo Administrators Guide* for more details on enabling SSH.

NOTE: *For Windows systems, AMX Shell requires a terminal client that supports SSH connections, such as PuTTY, TeraTerm, or Indigo Terminal Emulator.*

Default Connection Settings and Access Credentials

To connect to the Enzo meeting presentation system, the IP address is needed. On connection, a diagnostics screen can be accessed on which the IP address, Ethernet switch status, etc. can be viewed on the connected video output. Perform the following steps to locate the IP address of the device. For further details on navigating the Enzo interface, refer to the *Enzo Instruction Manual*.

1. On the Enzo opening screen, select **Start New Session**.
2. Perform a long mouse click on the **enzo** icon on the bottom right corner of the screen.
3. When the settings screen appears, click **System Settings**.
4. Select **Device Info**. The IP address of the Enzo unit appears in the Device area (see FIG. 1). The Device Info screen lists information about the device such as its IP address, firmware version, Enzo's available memory and the connected NetLinx Master, if any. Information in this area is view-only.

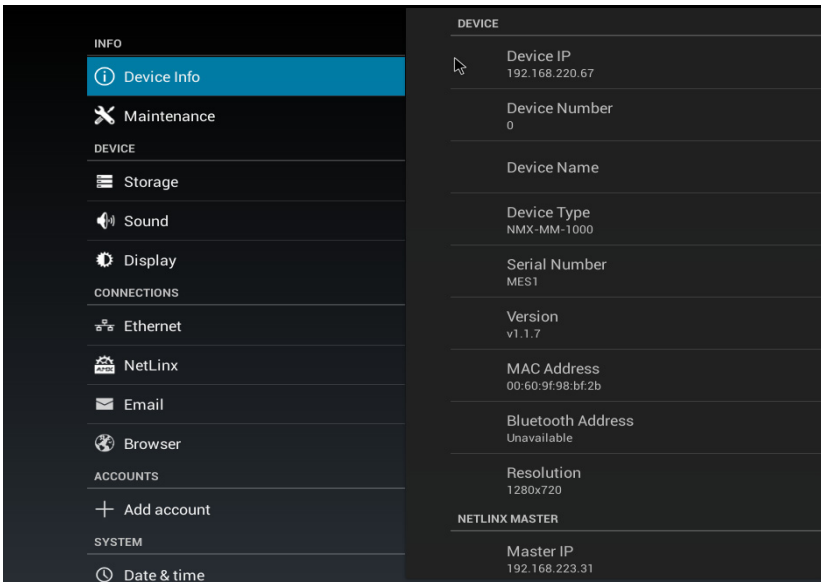


FIG. 1 Device Info Screen

The following table lists the default connection settings and access credentials for AMX Shell:

Connection Settings and Access Credentials	
Protocol	SSH
Username	amx
Password	1988 (This value will match the Security settings.)

Connecting to Enzo via AMX Shell

The following is an example of an SSH connection command:

```
> ssh amx@192.168.0.1 -p22 [ENTER]
```

In the above example, amx serves as the user name. The default port number is 22. Use the -p attribute to connect to a different port number. After pressing ENTER, type the password at the password prompt. (The default password is *1988*.)

Using a Windows Terminal Client to Connect to Enzo via AMX Shell

The terminal client used to connect to Enzo must support SSH connections. The log on process varies depending on the terminal client being used to connect to Enzo. Consult the terminal client documentation for help.

1. Open the terminal client and type the IP address of Enzo in the Hostname or IP Address text box.
2. Set the connection type to **SSH**.
3. When prompted for the user name, enter *amx*.
4. When prompted for the password, enter *1988*. The AMX Shell splash page appears (FIG. 2).

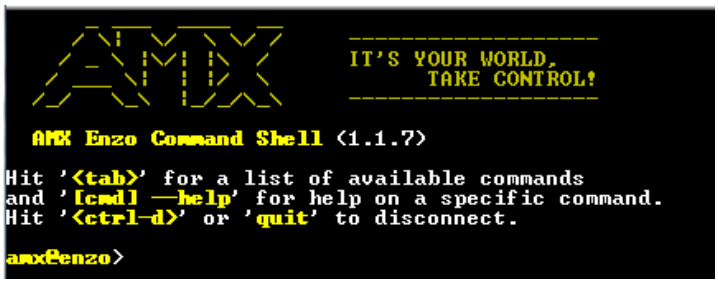


FIG. 2 AMX Shell Splash Page

Shell Commands

The following sections list the shell commands available for the Enzo device.

Command Auto-Complete

AMX Shell supports command auto-complete using the TAB key. Press the tab key at the command prompt anytime to see a listing of all available commands (FIG. 3).

```
> [TAB]
```

FIG. 3 Command List via Tab Key

Press the tab key after entering one or more characters to see the listing of possible command matches (FIG. 4).

```
> t [TAB]
```

FIG. 4 Command Matching with the Tab Key

If the auto-complete results find only a single match, the command is completed after the characters entered (FIG. 5).

```
> to [TAB]
```

FIG. 5 Auto-complete Using the Tab Key

Command Scopes

AMX Shell registers all commands with a "scope" attribute. The scope is an additional qualifier that can help distinguish duplicate command names at run/execution time. (Duplicate commands using the same command name and scope names are not permitted.)

NOTE: *It is best to avoid duplicate command names, but if necessary, use the scope to uniquely resolve each distinct command.*

NOTE: *Command names are also displayed without the scope identifier for commands that can be resolved without the scope identifier.*

The command scope is included in the command listing.

Use the scope name and press the TAB key to auto-complete and see the scope-specific command listing (FIG. 6).

```
> enzo [TAB]
```

```
amx@enzo>enzo:
enzo:about      enzo:alert      enzo:audio      enzo:blank
enzo:docmgr     enzo:keypad     enzo:mail       enzo:qr
enzo:session    enzo:video      enzo:webapp     enzo:webu
amx@enzo>enzo:
```

FIG. 6 Scope-specific Command Listing

A command can be executed with the fully-qualified command scope and command name (FIG. 7).

NOTE: *Full command scope is typically not required; however, if there are duplicate command names, include the scope to ensure that thee correct command is being executed.*

```
> enzo:keypad --close [ENTER]
```

```
amx@enzo>keypad --close
Closing virtual keypad controller screen.
amx@enzo>
```

FIG. 7 Full command scope

While in a user-specified scope, the scope name is included in the command prompt: `amx@Enzo(enzo)>`.

```
> scope enzo [ENTER]
```

```
amx@enzo>scope enzo
Current shell scope: enzo
amx@enzo(enzo)>
```

FIG. 8 User-specified Scope

While in a user-specified scope, the auto-complete listing is limited (filtered) to the commands available in the specified scope.

```
> scope enzo [ENTER]
> [TAB]\
```

```
amx@enzo(enzo)>
about       alert       audio       blank       docmgr      keypad
mail        qr          session     video       webapp      webu
amx@enzo(enzo)>
```

FIG. 9 User-specified Scope Commands

While in a user specified scope, the help command listing is limited (filtered) to the commands available in the specified scope.

```
> scope enzo [ENTER]
> help [ENTER]
```

```
amx@enzo(enzo)>scope enzo
Current shell scope: enzo
amx@enzo(enzo)>help

COMMAND LISTING

enzo:about          Display the about screen on the display device connect
enzo:alert          Display an alert notification on screen.
enzo:audio          Audio settings.
enzo:blank          Show/hide blank screen.
enzo:docmgr         Display the document management screen.
enzo:keypad         Display the virtual keypad controller on screen.
enzo:mail           Configure mail settings.
enzo:qr             Display a QR code.
enzo:session        Start or end an enzo session.
enzo:video          Video settings.
enzo:webapp         Open a web application URL in full screen view.
enzo:webu           Web Update
```

FIG. 10 User-specified Scope Command Listing

System Scope

All AMX Shell system commands are registered with the scope identifier "*" (asterisk). System-scoped commands are given the highest priority for command resolution.

Scope Command

The scope command is provided as a default system command. Use the scope command to specify a scope at runtime in which to work. If working in a specific scope, all commands executed via the shell will have biased command resolution to the selected scope.

The following command displays the scope command usage in the shell session.

```
> scope --help [ENTER]
```

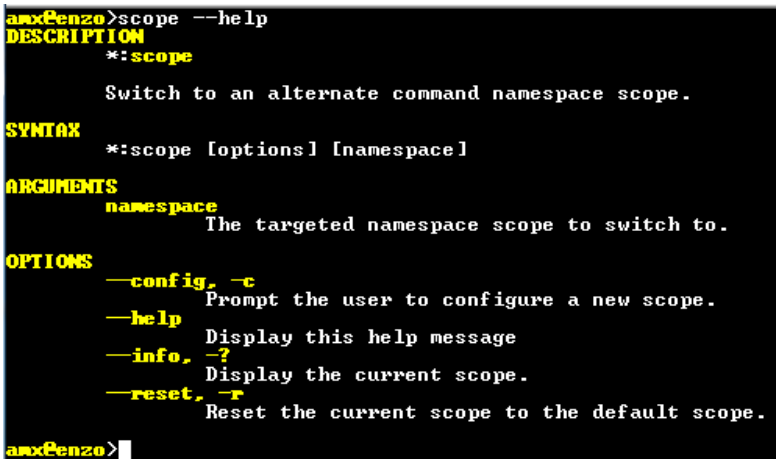


FIG. 11 Scope Command Help

Use the scope command to switch scopes (FIG. 12).

```
> scope [ENTER]
```

FIG. 12 Scope Command

Immediately switch to a new specified scope by including the scope namespace as an argument to the scope command.

```
> scope enzo [ENTER]
```

FIG. 13 Scope Command with Namespace

Return to the default system scope by using the "–reset" (or "-r") command option.

```
> scope --reset [ENTER]
```

FIG. 14 Reset Scope

Command Arguments

AMX Shell commands support command arguments. Command arguments typically include informational data that is required for the command to fulfill its intended goals.

The following command is an example of using a command argument to send a text message to the toast command. The toast command is displayed on the output connected to the Enzo.

```
> toast "This is a test of the emergency broadcast system" [ENTER]
```

NOTE: *If a command argument datum includes a space character, then the command argument string must be wrapped with double quotes.*

Command Options

AMX Shell supports command options. Command options always start with a hyphen ("-") character. Command options are used to provide additional (and often optional) content or instructions to the command execution.

NOTE: *Long option names start with two hyphen characters. Example: "--system" Abbreviated (short) option names start with a single hyphen character. Example: "-s".*

The following command is an example of using command options to apply specific settings values to the NetLinx command.

```
> NetLinx --system 0 --device 10005 --mode URL
```

Help Commands

List registered shell commands on-screen using the "help" command. The "?" command is an alias for "help".

```
> help [ENTER]
> ? [ENTER]
```

```
set             support         temp            time            toast
version         video           volume          webapp          webu
amxEenzo>help

COMMAND LISTING

?                       Displays this help or help about a command
back                    Issue the 'BACK' keystroke to the system.
clear                   Clears the console buffer.
date                    Gets/sets the current system date.
echo                    Echoes or prints arguments to STDOUT.
enzo:about              Display the about screen on the display device connect
enzo:alert              Display an alert notification on screen.
enzo:audio              Audio settings.
enzo:blank              Show/hide blank screen.
enzo:docmgr             Display the document management screen.
enzo:keypad             Display the virtual keypad controller on screen.
enzo:mail               Configure mail settings.
enzo:qr                 Display a QR code.
enzo:session            Start or end an enzo session.
enzo:video              Video settings.
enzo:webapp             Open a web application URL in full screen view.
enzo:webu               Web Update
exit                    Terminate the command shell session.
get                     Get information about a specific target provided as an
help                    Displays this help or help about a command
history                 Prints command history.
home                    Navigate 'HOME' on the system.
ip                      Gets/sets the IP settings of the device.
key                     Issue a keystroke to the system.
logout                  Terminate the command shell session.
man                     Displays this help or help about a command
msg                     Enable/disable diagnostics message logging.
netlinx                 Gets/sets the NetLinx ICSP connnection settings.
open                    Open URL in web browser. (i.e. http(s)://(url-address))
ping                    Test TCP/IP network connectivity with another device.
quit                    Terminate the command shell session.
reboot                  Reboot the device.
scope                   Switch to an alternate command namespace scope.
set                     Set the configuration for a specific target provided a
support                 Support utility command. Allows capturing of system ru
temp                    Report the device temperature in Celsius.
time                    Gets/sets the current system time.
toast                   Display a toast notification on the system.
version                 Display application version information.
volume                  Change the system volume level.

amxEenzo>
```

FIG. 15 Help Command

To access the usage details, command arguments, and options, include the "–help" option at the end of any command.

NOTE: *The "man" (manual) command can display the same command detail & usage information.*

> echo --help [ENTER]

```
> man echo [ENTER]
```

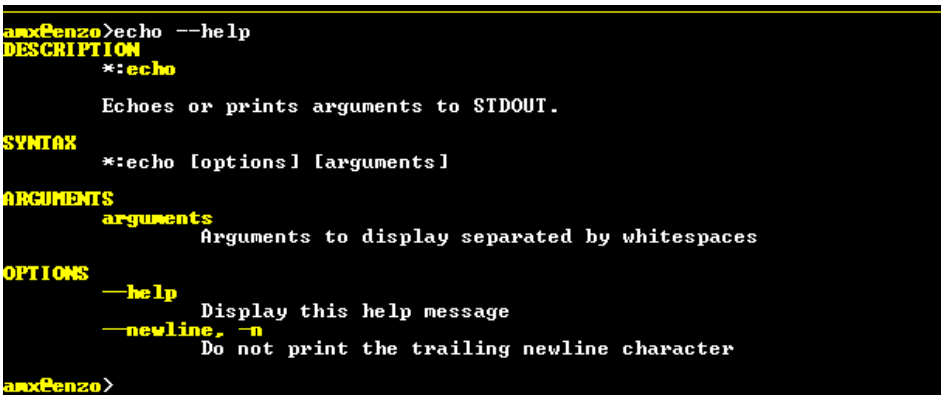


FIG. 16 --help option

Get/Set Command Proxies

Amx Shell defines specialized "get" and "set" commands to help establish a convention across all implementing products so that a common command syntax/notation for obtaining (get) information or applying (set) configuration settings.

The "get" and "set" commands are considered command proxies because all they do is execute the targeted underlying command with a prefixed command line option. The target command must support the command line options for the get and set command to function with the target command. Support for get and set commands is noted within each command definition in the command list tables below.

NOTE: *All shell commands that can provide status or display data support the --info command line option and thus support the get command.*

NOTE: *All shell commands that can be used to configure settings or apply runtime configuration support the --config command line option and thus support the set command.*

System Shell Commands

The following table lists the system shell commands. These commands are configured system commands and are available to all implementations of AMX Shell.

System Shell Commands	
clear	Clears the console buffer. Syntax: `*:clear`
date	Gets/sets the current system date. Syntax: `*:date [options] [date]` Arguments: date - New date in format: YYYY-MM-DD Options: --day, -d Day of month (1-31) (defaults to -1) --config, -c, --set Set the system date --help Display a help message --verbose, -v Display verbose date information --info, -? Display the current date on screen --month, -m Month (1-12) (defaults to -1) --year, -y Year (XXXX) (defaults to -1)

Continued ↴

System Shell Commands	
echo	Echoes the provided argument text. Syntax: `*:echo [options] [arguments]` Arguments: arguments - Arguments to display separated by whitespaces Options: --help Display a help message --newline, -n Do not print the trailing newline character
get	Command proxy used to get information about a specific target (command). See the *Get/Set Command Proxies* section on page 6 for more information. Syntax: `*:get [arguments]` Arguments: arguments - Command arguments to pass through.
help	Displays a help listing of all available shell commands to which the user has permission to access or displays help about a specific command. Syntax: `*:help [command]` Arguments: command - The command of which you wish to display the help information.
history	Prints the command history. Syntax: `*:history` **NOTE:** *Use the exclamation character followed by the command history index number to execute the command from history.* `Example: > !2 [ENTER]`
ip	Gets/sets the IP settings for the Enzo device. Syntax: `*:ip [options]` Options: --config, -c, --set Configure the set-up info interactively --dns1, -d1 The IP address of the primary DNS server --dns2, -d2 The IP address of the secondary DNS server --domain, -dn The domain name for the network --gateway, -gw The gateway IP address --help Displays a help message --hostname, -hn The host name for the device. (Alpha-numeric values, dashes, and no spaces.) --info, -i Displays the current IP settings --ipaddress, -ip The static IP address for the device. --mode, -m Set the connection mode (DHCP or Static) --reset, -r Reset IP settings to the factory default --subnetmask, -sn The subnet mask for the device
logout	Exits the shell and terminates the user's connection. (Same as "quit" or "exit") Syntax: `*:logout`

Continued ↴

System Shell Commands	
man	Displays detailed usage and CLI arguments and options information for a given command. Syntax: `*:man [command]` Arguments: command - The command for which to get help
msg	Enable/disable diagnostics message logging. Syntax: `*:msg [options] [instruction] [filters]` Arguments: instruction - Diagnostics message command instruction 'on': enable diagnostics messages 'off': disable diagnostics messages 'filter': sets optional log filters (provided by filters argument) 'add': add optional log filters (provided by filters argument) 'remove': removed optional log filters (provided by filters argument) 'clear': clear optional log filters 'delete': deletes current log filters - Optional log message filters (separated by spaces) Options: --add-filter, -af Add a filter to the current diagnostics log filters --clear-filter, -cf Remove all filters from diagnostics logging --clear-history, -ch, -d Delete the diagnostics log history --config, -c, --set Enable/disable diagnostics message output --filter, -f Optional log message filter --help Display a help message --info, -? Display current diagnostic message output status --off, -F, --disable, --stop Disable diagnostics message output --on, -N, --enable, --start Enable diagnostics message output --remove-filter, -rf Remove one or more filters from the current diagnostics log filter --show-filter, -sf Display all existing filters applied to diagnostics logging --verbose, -v Display verbose diagnostics message status information
ping	Test TCP/IP network connectivity with another device. Syntax: `*:ping [options] address` Arguments: address - IP address or URL Options: --help Display a help message --timeout, -w Timeout wait (number of seconds to wait for a response) --retry-count, -c Retry count (number of packets)
quit	Exits the shell and terminates the user's connection. (Same as "logout") Syntax: `*:quit`

Continued ↴

System Shell Commands	
reboot	Reboot the system. Syntax: `*:reboot [options]` Options: --silent, -s, -Y Do not prompt for confirmation; proceed with reboot. --help Display a help message
scope	Sets the current runtime command scope. While the session is in a command scope, commands that belong to this scope will be prioritized for runtime resolution. Executing this command with no arguments will return the shell to the default system scope. Syntax: `*:scope [options] [namespace]` Arguments: namespace - The targeted namespace scope to switch Options: --config, -c Prompt the user to configure a new scope --help Display a help message --info, -? Display the current scope --reset, -r Reset the current scope to the default scope
set	Command proxy used to set configuration on a specific target (command). See the *Get/Set Command Proxies* section on page 6 for more information. Syntax: `*:set arguments` Arguments: arguments - Command arguments to pass through.

Continued ↴

System Shell Commands	
time	Gets/sets the current system time. Syntax: `*:time [options] [time] [ampm]` Arguments: time - New time in format: 00:00:00 ampm - AM or PM (not needed if using 24 hour format) Options: --second, -s Second (0-59) (defaults to -1) --hour, -h Hour (0-24) (defaults to -1) --help Display a help message --info, -? Display the current time on screen. --am, -am AM (used when setting time) --minute, -m Minute (0-59) (defaults to -1) --millisecond, -ms Millisecond (0-999). (defaults to -1) --config, -c, --set Set the system time. --pm, -pm PM (used when setting time) --verbose, -v Display verbose time information.

Enzo Shell Commands

The following table lists the Enzo shell commands.

Enzo Shell Commands	
about	This command is used to open the "about" page in the Enzo settings GUI. Syntax: `enzo:about [options]` Options: --help Display this help message
alert	This command is used to submit alert notifications to the Enzo user interface. Include quotation marks (") around the title or message if using spaces in the text. This command supports the "set" command proxy. The "set alert" and "alert --config" commands invoke the interactive configuration mode. Syntax: `enzo:alert [options] [message]` Arguments: message - The message text for the alert notification. Options: --close, -c This option will close the alert notification dialog. (This option is exclusive. Any other options provided are ignored.) --config, -c, --set Configure the alert notification interactively. --help Display a help message --modal, -m Modal status for the alert notification. --timeout, -to Timeout in seconds for the alert notification. (The default is 30.) --title, -s, --subject The title text for the alert notification. --type, -t The message type for the alert notification (information, question, warning, security, critical). The default is information.
audio	This command is used to configure the audio output port selection. This command supports the "get" and "set" command proxies. The "set audio" and "audio --config" commands invoke the interactive configuration mode. Syntax: `enzo:audio [options]` Options: --beep, -b Test the audio output using a beep sound. --config, -c, --set Set the audio output port. --help Display a help message --info, -? Display the current audio output port. --out, -o Set audio output (HDMI, Analog). --verbose, -v Include detailed/verbose information about audio settings.

Continued ↴

Enzo Shell Commands	
blank	This command is used to display/hide the blanking screen in the Enzo user interface. This command supports the "set" command proxy. The "set blank" and "blank --config" commands will invoke the interactive configuration mode. Syntax: `enzo:blank [options] [visible]` Arguments: visible - Controls blanking visibility. 'on', 'show': show blank screen 'logo': show blank screen with logo 'off' 'hide': hide blank screen Options: --config, -c, --set Choose to show/hide the blanking screen from options menu. --help Display this help message --logo, -l Show blank screen with the AMX logo. --off, -f, --hide Hide blank screen. --on, -n, --show Show blank screen.
docmgr	Displays the document management screen. Syntax: `enzo:docmgr [options]` Options: --close, -x This option closes the document management screen. --help Display a help message
keypad	Displays the virtual keypad controller on screen. Syntax: `enzo:keypad [options]` Options: --close, -x This option closes the virtual keypad controller. --help Display a help message
mail	This command is used to display and/or configure SMTP mail settings. Syntax: `enzo:mail [options]` Options: --authentication, -a Set the authentication (ssl, tls, none) --from, -f Set the return address of the email --help Display a help message --host, -h Set the URL of the SMTP host --password, -p Set the password --port, -P Set the port number --test, -t Set a test email --user, -u Set the user name

Continued ↴

Enzo Shell Commands	
qr	Display a QR code. Syntax: `enzo:qr [options] [uri]` Arguments: uri - The URI to encode Options: --close, -x Close any open QR activity --link, -l This option displays the URI in text. --help Display a help message --title, -t This option places a title on the page.
session	Start or end an Enzo session. Syntax: `enzo:session [options] [action]` Arguments: action - start or end action to perform. 'start': start session 'end', 'stop': end session Options: --config, -c, --set Choose to start/end a session from options menu --end, -e, --stop End current session (if in a session) --help Display a help message --info, -? Display the current session state --start, -s Start a new session (if not already in a session)
video	Sets the video settings. Syntax: `enzo:video [options]` Options: --config, -c, --set Set the video output resolution settings --help Display a help message --info, -? Display the current video output resolution settings --out, -o Set the display video resolution (1080p, 720p, EDID) --reboot, -r Reboot the system after applying the new video resolution

Continued ↴

Enzo Shell Commands	
webapp	Open a web application URL in full screen view. Syntax: `enzo:webapp [options] url` Arguments: url - The URL address to open Options: --close, -x This option closes the Web browser --help Display a help message --no-scheme, -ns Do not apply a default scheme (protocol prefix) to the provided URL --title, -t The title screen for this Web page.
webu	Perform a web update where the new firmware kit file is provided by the specified URL. Syntax: `enzo:webu [options] url` Arguments: url - The URL address to open Options: --help Display a help message

NetLinx Programming

Overview

This chapter defines all programming commands and system responses available for the NMX-MM-1000 Enzo Meeting Presentation System.

Device Ports:

The following table details the device ports on an Enzo system:

Port	Name	Description
1	Enzo	API exposed on this device port controls the core Enzo platform, settings, and Enzo operating environment.
2	Virtual Keypad	API exposed on this device port controls/interacts with the Virtual Keypad implementation built into Enzo's user interface.
3	Serial (RS-232)	API exposed on this device port controls/interacts with the serial port (RS-232) on the rear panel of the Enzo hardware. Port 3 of the Enzo ICSP device is used for communication and configuration of Enzo's serial port from a connected NetLinx master. **Note:** *As a best practice the baud rate for the serial port should be set when Enzo comes online.*

SEND_COMMANDS

The commands listed in the following sections are Enzo specific. For generic NetLinx commands, see the *NetLinx Integrated Controllers WebConsole and Programming Guide* available at *http://www.amx.com*.

- The commands derive their input/output port addressing from the target D:P:S.
- Input and Output functional distinctions are disambiguated from the overlapped port numbers by combining them with the command name.

The following table lists the channels for the Enzo:

Channels		
Channel	**Name**	**Description**
1	Play	Issues a PLAY command to the active app. Actual results will be app specific.
2	Stop	Issues a STOP command to the active app. Actual results will be app specific.
3	Pause	Issues a PAUSE command to the active app. Actual results will be app specific.
4	Next	Issues a NEXT command to the active app. Actual results will be app specific.
5	Previous	Issues a PREVIOUS command to the active app. Actual results will be app specific.
6	Fast Forward	Issues a FAST FORWARD command to the active app. Actual results will be app specific.
7	Rewind	Issues a REWIND command to the active app. Actual results will be app specific.
24	Volume Up	Ramps the volume up on the attached display.
25	Volume Down	Ramps the volume down on the attached display.
26	Mute Cycle	Cycles the volume mute on the attached display.

NetLinx Commands

The following list of commands may all be executed using the NetLinx SEND_COMMAND command. Commands with **AUTOSTART** in the description fields will execute whether Enzo is in an active session or not.1988 Issuing this command will automatically start a session and launch the command.

NetLinx Commands	Description
ALERT	Displays an alert message. Of the arguments to pass with this command, only message is required. All other arguments are optional. Syntax: `SEND_COMMAND <DEV>, "'ALERT-<message>'"` `SEND_COMMAND <DEV>, "'ALERT-<message>,<type>'"` `SEND_COMMAND <DEV>, "'ALERT-<message>,<type>,<title>'"` `SEND_COMMAND <DEV>, "'ALERT-<message>,<type>,<title>,<modal>'"` `SEND_COMMAND <DEV>, "'ALERT-<message>,<type>,<title>,<modal>,<timeout>'"` Variables: *message* = The message to send *type* = The type of alert (optional). Accepted values are 'information', 'warning', 'question', 'security', and 'critical'. *title* = The title of the alert (optional). Suggested length is 60 characters or less, no hard limit is set. *modal* = The modal status for the alert (true or false), (optional). Default is *false*. *timeout* = The timeout in seconds for the alert message (optional). Default is *30*. Example: `SEND_COMMAND 10005:1:0, "'ALERT-Exit Building Now, critical,, true'"`
ALERT.CLOSE	Closes any active alert message. Syntax: `SEND_COMMAND <DEV>, "'ALERT.CLOSE'"`
APP.LAUNCH	Launches supported applications included with Enzo. **AUTOSTART** Syntax: `SEND_COMMAND <DEV>, "'APP.LAUNCH-<application>'"` Variables: MIRROROP, BROWSER, FIREFOX and TV are currently the only supported applications. **Note:** *On browser launch, you may get a notification "Older version of chrome is detected" and an offer to update to stay secure. Please ignore this response since the browser is embedded on Enzo and cannot be upgraded through the browser app.* Example: `SEND_COMMAND 10005:1:0, "'APP.LAUNCH-MIRROROP'"` Response: None.
APP.MODE	This command sets the Enzo launch screen to display the normal home screen with all Apps listed, or configures Enzo to open a session only in an app selected by the administrator. Syntax: APP.MODE-<mode> Variables: *MULTI* = for standard multiple app mode. The Enzo main screen should show after changing to multiple app mode, and allow for any app to launch. *"App Name"* = for single app mode. A new session will only see the app specified by the Administrator. The simple App Name is the same as the one used for APP.LAUNCH. Currently supported are TV, MirrorOp, and Browser Example: `APP.MODE-Browser` **IMPORTANT:** *Setting the App Mode purges session data.*
?APP.MODE	Returns the app mode info which includes: • "*App name*" for single app mode (TV, MirrorOp, or Browser) or *"MULTI"* for multiple app mode • The App Mode Purge value of Yes, No, or Interactive. Please see the APP.MODE_PURGE for more info. Syntax: `?APP.MODE`

Continued ↴

NetLinx Commands	Description
APP.MODE_PURGE	This command determines the Purge on Home and Escape settings that determine what happens when the user presses Home or Escape (ESC) on the keyboard. **IMPORTANT:** *This command has no impact while in multi app mode.* When in single app mode, the admin can access the system settings menu by one of the following methods: 1. Press and hold the ID button on the front of the Enzo unit. This will bring up a Setting dialog that will allow entering settings by selecting it with a keyboard or mouse. 2. Press Escape while on the Session Confirm Activity and show *Settings* by pressing F1 or long click the Enzo logo. 3. After the admin is done, press Home to show the Session Confirm Activity again. In the other to modes the admin may change to Multi App Mode in the Web UI to make Setting accessible on Enzo again. Syntax: `APP.MODE_PURGE-<mode>` Variables: • *Yes* - Pressing escape or home in single app mode will purge and resume the single app mode app. • *No* - Pressing escape or home in single app mode will only pause and resume the single app mode app without purging. • *Interactive* - Pressing escape or home will pause the single app mode app and show a user interactive dialog providing the user with the option to purge or not purge. **Sleep Warning:** *If the Single AppMode App holds a wake lock then Enzo will not sleep. If the user shows Session Confirm Activity and the Enzo times out for the Display timeout then the Single AppMode App will restart.*
BACK	Issues the BACK command to the operating system. Syntax: `SEND_COMMAND <DEV>, "'BACK'"`
BLANK	Starts an activity that shows a black screen known as the "blanking screen". Syntax: `SEND_COMMAND <DEV>, "'BLANK'"`
BLANK.CLOSE	Closes any active blanking activity display. Syntax: `SEND_COMMAND <DEV>, "'BLANK.CLOSE'"`
BLANK.LOGO	Displays the blanking screen with the optional animated AMX spinning logo. Syntax: `SEND_COMMAND <DEV>, "'BLANK.LOGO'"`
BROWSER	Opens the default web browser starting at the default home page. The browser options are Firefox (default) and Browser. **AUTOSTART** Variables • url - the URI to display • title (optional) - the title of the display (Currently Firefox only shows this while loading the page.) • mobile (optional) - true|false : 'true' will request the URL to load the mobile version of the web page. 'false' will request the URL to load the desktop version of the web page. (Note: If this option is omitted, it defaults to 'false' thus resulting in a 'desktop' version of the requested URL.) Syntax: `SEND_COMMAND <dev>,'BROWSER-url,title,true'`

Continued ↴

NetLinx Commands	Description
CONTENT.ACTION.EMAIL	E-mail the selected item/file as an attachment to a list of recipient e-mail addresses. Syntax: `SEND_COMMAND <DEV>,"'CONTENT.ACTION.EMAIL-<item-key>,<email-to>,<subject>,<message>` Variables: <*item-key*> = (required) The desired item path to email. <*email-to*> = (required) Email (TO) address list. (semicolon delimited) <*subject*> (optional) = Email subject text. (Optional) <*message*> (optional) =Email message/body text. (Optional) Examples: `SEND_COMMAND 10005:0:1,"'CONTENT.ACTION.EMAIL-/folder/sub-folder/content-item.xxx,user@domain.com'"` `SEND_COMMAND 10005:0:1,"'CONTENT.ACTION.EMAIL-/folder/sub-folder/content-item.xxx,user1@domain.com;user2@domain.com'"` `SEND_COMMAND 10005:0:1,"'CONTENT.ACTION.EMAIL-/folder/sub-folder/content-item.xxx,user@domain.com,subject'"` `SEND_COMMAND 10005:0:1,"'CONTENT.ACTION.EMAIL-/folder/sub-folder/content-item.xxx,user@domain.com,subject,message'"` Response: `CONTENT.ACTION.EMAIL.SUCCESS-<item-path>` **Note:** *If any error is encountered while trying to email the content item, the following error event notification will be broadcast.* `CONTENT.ACTION.EMAIL.ERROR-<error-message>,<item-path>`
CONTENT.ACTION.OPEN	Open the selected item/file using the default viewing application. Syntax: `SEND_COMMAND <DEV>, "'CONTENT.ACTION.OPEN-<item-path>'"` Variable: <*item-path*> = (required) The desired item path to open Example: `SEND_COMMAND 10005:0:1,"'CONTENT.ACTION.OPEN-/folder/sub-folder/content-item.xxx'"` Response: `CONTENT.ACTION.OPEN.SUCCESS-<item-path>` **Note:** *If any error is encountered while trying to open the content item, the following error event notification will be broadcast.* `CONTENT.ACTION.OPEN.ERROR-<error-message>,<item-path>`
CONTENT.ACTION.SHARE	Share the selected item/file if the content source supports sharing. Syntax: `SEND_COMMAND <DEV>, "'CONTENT.ACTION.SHARE-<item-key>'"` Variable: <*item-key*> = (required) The desired item path to share Example: `SEND_COMMAND 10005:0:1,"'CONTENT.ACTION.SHARE-/folder/sub-folder/content-item.xxx'"` Response: `CONTENT.ACTION.SHARE.SUCCESS-<item-path>` **Note:** *If any error is encountered while trying to share the content item, the following error event notification will be broadcast.* `CONTENT.ACTION.SHARE.ERROR-<error-message>,<item-path>`

Continued ↴

NetLinx Commands	Description
?CONTENT.ITEM	Retrieve the details of a specific content item available in the current content cursor path and source. Syntax: `SEND_COMMAND <DEV>, "'?CONTENT.ITEM-<item-path>'"` Variable: *item-path* = (required) The desired item to query. Example: `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEM-/folder/sub-folder/content-item.xxx'"` Response: `CONTENT.ITEM-<source-id>, <item-path>, <item-name>, <item-type>, <item-size>, <item-last-modified>, <item-read-only>` **Note:** *If any error is encountered while trying to query the content item, the following error event notification will be broadcast.* `CONTENT.ITEM.ERROR-<error-message>,<item-path>`
?CONTENT.ITEMS	Retrieve the listing of items available via the current content cursor path and source. Syntax: `SEND_COMMAND <DEV>, "'?CONTENT.ITEMS-(<start-index>), (<records-count>),(<exclude-special-items:true|false>)"` Variables: *start-index* (optional) = The starting record to return in the results response. If not provided, the result set will start with the first index. (The index is one-based; not zero-based.) If the starting index is greater than the total available records, then no records will be returned in the response. *records-count* (optional) = The number of records to return in the results response. If not provided, the result set will include all records to the end of the result set. (If the requested record count is greater than the total available records remaining, then only the available records to the end of the listing will be returned in the response.) *exclude-special-items* (optional)= Instruct the item records returned to include (value = 'false') or exclude (value = 'true') special UI only items such as <Clear Search Results> and <Up To Parent> items in the result set. If this argument is not provided, the results items will include the special items. Examples: `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS-4'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS-10,30'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS'-10,30,true"` `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS'-,,true"` Response: `CONTENT.ITEMS.RECORD.COUNT-<relative-records-count>,<absolute-records-count>` **Note:** *If the number of content item (records) is greater than zero, then the responses will include the following command event for each content item record.* `CONTENT.ITEMS.RECORD-<relative-record-index>,<absolute-record-index>,<source-id>,<item-path>,<item-name>,<item-type>,<item-size>,<item-last-modified>,<item-read-only>` **Note:** *If any error is encountered while trying to query the list of content items, the following error event notification will be broadcast.* `CONTENT.ITEMS.ERROR-<error-message>`

Continued ↴

NetLinx Commands	Description
?CONTENT.ITEMS.COUNT	Retrieve the total number of records available in the current content listing. Syntax: `SEND_COMMAND <DEV>,"'?CONTENT.ITEMS.COUNT-<exclude-special-items:true|false'"` Variable: *exclude-special-items* (optional) = Instruct the item records returned to include (value = 'false') or exclude (value = 'true') special UI only items such as <Clear Search Results> and <Up To Parent> items in the result set. If this argument is not provided, the results items will include the special items. Examples: `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS.COUNT'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS.COUNT-true'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.ITEMS.COUNT-false'"` Response: `CONTENT.ITEMS.COUNT-<records-count>` **Note:** *If any error is encountered while trying to query the total record count from the content items, the following error event notification will be broadcast.* `CONTENT.ITEMS.COUNT.ERROR-<error-message>`
?CONTENT.PATH	Retrieve the current working path in the content cursor for the current selected content source. Syntax: `SEND_COMMAND <DEV>,"'?CONTENT.PATH'"` Example: `SEND_COMMAND 10005:0:1,"'?CONTENT.PATH'"` Response: A confirmation event will be returned including the current source and content cursor path. `CONTENT.PATH-<source-id>,<path>` **Note:** *If the requested path is not found or any other error is encountered while trying to assign the content cursor path, the following error event notification will be broadcast.* `CONTENT.PATH.ERROR-<error-message>,<current-path>`
CONTENT.PATH	Assign/select a working path in the content cursor for the current selected content source. Syntax: `SEND_COMMAND <DEV>, "'CONTENT.PATH-<path>'"` Variable: *path (optional)* = The desired path to assign to the content cursor. If omitted or empty, the root path will be applied for the content source. The value of "{ROOT}" can be specified to select the root path for the current content cursor source. The value of "{UP}" can be specified to select the parent path for the current content cursor path. The value of "{REFRESH}" can be specified to refresh last known path for the current content cursor path The value of "{SEARCH.CLEAR}" can be specified to select the last known path for the current content cursor path thus clearing the search result listing. Examples: `SEND_COMMAND 10005:0:1,"'CONTENT.PATH'"` `SEND_COMMAND 10005:0:1,"'CONTENT.PATH-{ROOT}'"` `SEND_COMMAND 10005:0:1,"'CONTENT.PATH-{UP}'"` `SEND_COMMAND 10005:0:1,"'CONTENT.PATH-{REFRESH}'"` `SEND_COMMAND 10005:0:1,"'CONTENT.PATH-{SEARCH.CLEAR}'"` `SEND_COMMAND 10005:0:1,"'CONTENT.PATH-/folder-name/sub-folder'"` Response: A confirmation event will be returned including the current source and content cursor path. `CONTENT.PATH-<source-id>,<path>` **Note:** *If the requested path is not found or any other error is encountered while trying to assign the content cursor path, the following error event notification will be broadcast.* `CONTENT.PATH.ERROR-<error-message>,<requested-path>`

Continued ↴

NetLinx Commands	Description
CONTENT.SEARCH	Retrieve the listing of content items available via the current content source that match the requested search term. The records matching the search criteria will be available in the content listing records. Use the "?CONTENT.ITEMS" command to query the item records after the search listing is available. Syntax: `SEND_COMMAND <DEV>, "'CONTENT.SEARCH-<search-term>'"` Variable: *search-term* = (required) The search term/expression used in the source content lookup. Examples: `SEND_COMMAND 10005:0:1,"'CONTENT.SEARCH-budget'"` `SEND_COMMAND 10005:0:1,"'CONTENT.SEARCH-annual'"` Response: `CONTENT.SEARCH.RESULT-<search-term>,<search-results-count>` **Note:** *If any error is encountered while trying to query the total record count from the content items, the following error event notification will be broadcast.* `CONTENT.SEARCH.ERROR-<error-message>,<search-term>`
CONTENT.SEARCH.CLEAR	Clear the search query result item listing and switch back to the last known content path listing. Syntax: `SEND_COMMAND <DEV>,"'CONTENT.SEARCH.CLEAR'"` Examples: `SEND_COMMAND 10005:0:1,"'?CONTENT.SEARCH.CLEAR-budget'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.SEARCH.CLEAR-annual'"` Response: A confirmation event will be returned including the current source and content cursor path. `CONTENT.PATH-<source-id>,<path>` **Note:** *If the requested path is not found or any other error is encountered while trying to assign the content cursor path, the following error event notification will be broadcast.* `CONTENT.PATH.ERROR-<error-message>,<current-path>`
CONTENT.SOURCE	Assign/select a content source to the content cursor. This source will be assigned as the current content source on the content cursor for future queries. Syntax: `SEND_COMMAND <DEV>, "'?CONTENT.SOURCE-<source-id>'"` Variable: *source-id* = (required) The content source to assign. *show navigator* (optional) : If true then it will show the Navigator Example: `SEND_COMMAND 10005:0:1, "'CONTENT.SOURCE-usb, true'"` Response: A confirmation event will be returned including the detailed properties of the newly assigned content source. `CONTENT.SOURCE.CHANGED-<source-id>,` `<source-name>,` `<source-root-path>,` `<source-is-ready:true|false>` **Note:** *If the requested path is not found or any other error is encountered while trying to assign the content cursor path, the following error event notification will be broadcast.* `CONTENT.SOURCE.ERROR-<error-message>,<requested-source-id>` If not in session then you will get the following error: `CONTENT.ERROR-The Enzo session is not active; please start a session first.`

Continued ↴

NetLinx Commands	Description
CONTENT.SOURCE. EJECT	Ejects a detachable content source. If an explicit content source ID is not provided in the request command, the current selected source on the content cursor is used. Syntax: `SEND_COMMAND <DEV>,"'CONTENT.SOURCE.EJECT-<source-id>'"` Variables: *source-id* = (optional) The content source to eject (optional) Examples: `SEND_COMMAND 10005:0:1,"'CONTENT.SOURCE.EJECT'"` `SEND_COMMAND 10005:0:1,"'CONTENT.SOURCE.EJECT-usb'"` Responses: `CONTENT.SOURCE.EJECTED-<source-id>` **Note:** *If the requested source is not found or if another error occurs, then the following error event notification will be broadcast.* `CONTENT.SOURCE.EJECT.ERROR-<error-message>,<requested-source-id>` **Note:** *If the affected source was the current selected source, then the `CONTENT.SOURCE` event may also be broadcast.*
CONTENT.SOURCE. LOGOUT	Log off the current content source. If an explicit content source ID is not provided in the request command, the current selected source on the content cursor is used. Syntax: `SEND_COMMAND <DEV>, "'CONTENT.SOURCE.LOGOUT-<source-id>'"` Variables: *source-id* = (optional) The content source to log out (optional) Examples: `SEND_COMMAND 10005:0:1,"'CONTENT.SOURCE.LOGOUT'"` `SEND_COMMAND 10005:0:1,"'CONTENT.SOURCE.LOGOUT-dropbox'"` Responses: `CONTENT.SOURCE.LOGOUT-<source-id>` **Note:** *If the requested source is not found or if another error occurs, then the following error event notification will be broadcast.* `CONTENT.SOURCE.LOGOUT.ERROR-<error-message>,<requested-source-id>` **Note:** *If the affected source was the current selected source, then the `CONTENT.SOURCE` event may also be broadcast.*
?CONTENT.SOURCE	Retrieve detailed properties of a content source. If an explicit content source ID is not provided in the request command, the current selected source on the content cursor is used. Syntax: `SEND_COMMAND <DEV>, "'?CONTENT.SOURCE-<source-id>'"` Variable: *source-id* = (optional) The content source from which to query properties (optional) Examples: `SEND_COMMAND 10005:0:1,"'?CONTENT.SOURCE'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.SOURCE-usb'"` Response: `CONTENT.SOURCE-<source-id>,` `<source-name>,` `<source-root-path>,` `<source-is-ready:true|false>` **Note:** *If the requested source is not found or if another error occurs, then the following error event notification will be broadcast.* `CONTENT.SOURCE.ERROR-<error-message>,<requested-source-id>`

Continued ↴

NetLinx Commands	Description
?CONTENT.SOURCES	Retrieve a listing of enabled content sources. Syntax: `SEND_COMMAND <DEV>,"'CONTENT.SOURCES-<start-index>,<records-count>'"` Variables: *start-index* (optional) = The starting record to return in the results response. If not provided, the result set will start with the first index. (index is one-based; not zero-based) If the starting index is greater than the total available records, then no records will be returned in the response. *records-count* (optional) = The number of records to return in the results response. If not provided, the result set will include all records to the end of the result set. (If the requested record count is greater than the total available records remaining, then only the available records to the end of the listing will be returned in the response.) Examples: `SEND_COMMAND 10005:0:1,"'?CONTENT.SOURCES'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.SOURCES-2'"` `SEND_COMMAND 10005:0:1,"'?CONTENT.SOURCES-2,5'"` Responses: `CONTENT.SOURCES.RECORD.COUNT-<relative-records-count>,<absolute-records-count>` **Note:** *If the number of content sources (records) is greater than zero, then the responses will include the following command event for each source record.* `CONTENT.SOURCES.RECORD-<relative-record-index>,` `<absolute-record-index>,` `<source-id>,` `<source-name>,` `<source-root-path>,` `<source-is-ready:true|false>` **Note:** *If any error is encountered while trying to query the list of content sources, the following error event notification will be broadcast.* `CONTENT.SOURCES.ERROR-<error-message>`
?CONTENT.SOURCES. COUNT	Query for the total number of content source records available. Syntax: `SEND_COMMAND <DEV>, "'?CONTENT.SOURCES.COUNT'"` Responses: `CONTENT.SOURCES.COUNT-<total-records-available>` **Note:** *If any error is encountered while trying to query the total record count from the content sources, the following error event notification will be broadcast.* `CONTENT.SOURCES.COUNT.ERROR-<error-message>`
DOWN	Issues an ARROW-DOWN keystroke to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'DOWN'"`
DOWNLOAD.ABORT	Causes current AMX content sharing transfer to be aborted. Syntax: `SEND_COMMAND 10005:0:1,"DOWNLOAD.ABORT"`
ENTER	Issues a ENTER keystroke to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'ENTER'"`
EXIT	Ends the current session. Syntax: `SEND_COMMAND <DEV>, "'EXIT'"`
FFWD	Issues a FAST FORWARD command to the active app. Actual results are app-specific. Syntax: `SEND_COMMAND <DEV>, "'FFWD'"`
GET BAUD	Get the RS-232/422/485 port's current communication parameters. The port sends the parameters to the device that requested the information. Syntax: `GET BAUD` Example: `SEND_COMMAND ENZO_RS232,"'GET BAUD'"` The port responds with: `PORT <port#>,<baud>,<parity>,<data>,<stop> 485 DISABLE` System response example: `PORT 3,38400,N,8,1 485 DISABLE`

Continued ↴

NetLinx Commands	Description
HOME	Issues the HOME command to the operating system. Syntax: `SEND_COMMAND <DEV>, "'HOME'"` Response: A Home state response will be issued any time the user returns to the Home screen. `ACTIVITY.STATE-ONSTARTING,HOME`
KEY	Issues a series of keystrokes to the active app. Syntax: `SEND_COMMAND <DEV>, "'KEY-<string>'"` Variable: *string* = The string of keystrokes to send. Example: `SEND_COMMAND 1001:1:0, "'KEY-Hello, World'"`
LEFT	Issues an ARROW-LEFT keystroke to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'LEFT'"`
?MIRROROP.CONNECTION	Returns the settings for the connection that you can change with MIRROROP.MANAGE. Syntax: `?MIRROROP.CONNECTION-<senderId>` Variable: *senderId* - The sender identifier or "All" with the response for each connection info request:
MIRROROP.CONTROL	Enable or disable senders ability to obtain a quadrant automatically. `MIRROROP.CONTROL-<senderId>,<true/false>` Variables: *senderId* - "All" for all senders *true/false* - Determines if the sender is allowed to obtain a quadrant automatically or not.
MIRROROP.DISCONNECT	Disconnect all senders from the MirrorOp app or stop individual senders from playing. `MIRROROP.DISCONNECT-<senderId>` Variable: *senderId* - The integer value of the sender or "all" to reset all connections. **Note:** *Individual disconnects actually just stop playback while "all" actually disconnects all the senders.*
MIRROROP.MANAGE	Assign a Sender to a specific quadrant or full screen. `MIRROROP.MANAGE-<senderId>,<display>,<quadrant>` Variables: *senderId* - The integer value associated with the specific connection. *display* - determines the sender screen that is to appear. 0 for primary or 1 more for other extended displays. (Only 0 is currently used as Senders are only allowed to send one display at a time.) *quadrant* - determines fullscreen with "fullscreen" or a quadrant with 1-4 or fullscreen. (Top Left = 1, Top Right = 2, Bottom Left = 3, Bottom Right = 4) **Note:** *A connection must be made before it can be managed unless it is manually added with the MIRROROP.CONTROL command.*
MIRROROP.STARTSCREEN	Used to set the MirrorOp start screen including the host prompt, code prompts and background image. MIRROROP.STARTSCREEN-<locale>,<receiver_name>,<subject>, <host_prompt>,<code_prompt>,<background_image> Syntax: SEND_COMMAND <DEV>, " 'MIRROROP.STARTSCREEN-en_US,Enzo Help Desk,,IP: $IP, Code: $PASSCODE' " Variables: locale - currently only en-US is supported. receiver_name - The name or title of the receiver and allows for $NAME which is the Name value from the About Screen in the Enzo Settings subject - currently blank and unused host_prompt - contains $IP keyword to display IP address on MirrorOp page code_prompt - contains the desired text for display and allows $PASSCODE keywords for replacement background_image - the path to a 720p or 1080p background image (optional) Note: Receiver name from the About Screen is limited to 30 characters or less. With $NAME the receiver name can be longer, but we still recommend 30 characters or less.

Continued ↴

NetLinx Commands	Description
MIRROROP.STARTSCREEN_ LAYOUT	Changes the layout to a selected one of the 6 options available. Syntax: `MIRROROP.STARTSCREEN_LAYOUT-<locale>,<layout>` Variables: *locale* - Allows any valid locale usually en_US. *layout* - valid values include: • *current* for Default • *top* for Top Justified • *top_left* for Top Left • *top_right* for Top Right • *top_middle* for Top Middle • *bottom_middle* for Bottom Middle
MIRROROP.STARTSCREEN_ RECEIVER_FONT	Changes the receiver font size & color. Syntax: `MIRROROP.STARTSCREEN_RECEIVER_FONT-<locale>,<font_size>,<font_color>` Variables: *locale* - Allows any valid locale usually en_US. *font size* - numeric value for the size of the font, only applies to the receiver name. Valid range is 1 through 400 with default value = 0 (Currently 36). *font color* - applies only to the receiver name. Use the WebApp to select from an infinite array of colors. Range for RGB 000000-FFFFFF or for ARGB 00000000-FFFFFFFF. Default is #636363 and represents dark gray.
MIRROROP.STARTSCREEN_ HOST_AND_CODE_FONT	Changes the host and code font size & color. Syntax: `MIRROROP.STARTSCREEN_HOST_AND_CODE_FONT-<locale>,<font_size>,<font_color>` Variables: *locale* - Allows any valid locale usually en_US. *font size* - numeric value for the size of the font, only applies to the receiver name. Valid range is 1 through 400 with default value = 0 (Currently 84). *font color* - applies to the host and code font. Use the WebApp to select from an infinite array of colors. Range for RGB 000000-FFFFFF or for ARGB 00000000-FFFFFFFF. Default is #636363 and represents dark gray.
NEXT	Issues a NEXT command to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'NEXT'"`
PAGE.DOWN	Issues a PAGE DOWN keystroke to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'PAGE.DOWN'"`
PAGE.UP	Issues a PAGE UP keystroke to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'PAGE.UP'"`
PASSCODE	Sets the Passcode for MirrorOp presenters. Syntax: `PASSCODE-<random/value/0>,true/false` Variables: *true* = default and keeps existing senders *false* = kicks existing senders off *'0'* = will set the MirrorOp passcode to no password required **Note:** *The Enzo web UI and NetLinx commands allow invalid passcodes to be set on Enzo. The valid passcode range is 1000 to 9999. Setting the passcode less than 1000 is equivalent to disabling the passcode. If an invalid passcode is set via the web UI, it will revert to the previously set valid passcode following an Enzo reboot.*
?PASSCODE	Retrieve the current session passcode. Syntax: `SEND_COMMAND <DEV>, "'?PASSCODE'"` Example: `SEND_COMMAND 10005:1:0,"'?PASSCODE'"` Responses: • No Session active = PASSCODE-NONE • Session active no passcode set = PASSCODE-OPEN • Session active passcode set = PASSCODE-<passcode>, PASSCODE-1234

Continued ↴

NetLinx Commands	Description
PAUSE	Issues a PAUSE command to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'PAUSE'"`
PLAY	Issues a PLAY command to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'PLAY'"`
PREVIOUS	Issues a PREVIOUS command to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'PREVIOUS'"`
QR	Displays a QR code on the video output. Syntax: `SEND_COMMAND <DEV>, "'QR-<url>'"` `SEND_COMMAND <DEV>, "'QR-<url>,<title>'"` `SEND_COMMAND <DEV>, "'QR-<url>,<title>,<link>'"` Variables: url = The actual URL to display as a QR code title = (optional) A title to display for the QR code. This argument is optional. link = (optional) true or false. Display a text version of the URL. Example: `SEND_COMMAND 1001:1:0,"'QR-http://www.amx.com,,true'"` Shows a QR code for the AMX website without a title but with a text version of the URL. **Note:** *Title suggested length is 60 characters or less. However, no hard limit is set.*
QR.CLOSE	Closes any active QR code display. Syntax: `SEND_COMMAND <DEV>, "'qr.CLOSE'"`
REWIND	Issues a REWIND command to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'REWIND'"`
RIGHT	Issues an ARROW-RIGHT keystroke to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'RIGHT'"`
?SESSION	Retrieve the current session state. Syntax: `SEND_COMMAND <dev>,'?SESSION'` Responses: If a new session is started: `SESSION-TRUE` If session is ended: `SESSION-FALSE` If Enzo is in single app mode and responds false to a ?SESSION command, this indicates that Enzo is in a timed out state as there is no other way to be out of session while in single app mode. Wake Enzo up with any number of commands other than APP.LAUNCH and put a delay to allow for the known multiple launches of the app after wake up, then issue the APP.LAUNCH for the app desired. Use the ACTIVITY.STATE responses to track the open app and reopen the desired app if needed. These session statuses are also unsolicited responses in NetLinx Studio when streaming is started or ended.
SET BAUD	Set the COM port's communication parameters. **Note:** *Before Release 8 this value was not saved in Non-Volatile memory, and is reset to default (9600,8,N,1) at power-up. For, release 8 this now persists between reboots. However, if you change to a device that does not support the current setting it may not connect..* Syntax: `SET BAUD <baud>,<parity>,<data>,<stop> [485 <Enable | Disable>]` Variables: • Baud: 115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 150. • Parity: N (none), O (odd), E (even) • Data: 7 or 8 data bits • Stop: 1 or 2 stop bits Example: `SEND_COMMAND ENZO_RS232,"'SET BAUD 19200,N,8,1'"` `SEND_COMMAND ENZO_RS232,"'SET BAUD 19200,N,8,1'"` Sets the ENZO_RS232 port's communication parameters to 19,200 baud, no parity, 8 data bits, and 1 stop bit.

Continued ↴

NetLinx Commands	Description
START	Starts a new session **AUTOSTART** Syntax: `SEND_COMMAND <DEV>, "'START'"`
STOP	Issues a STOP command to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'STOP'"`
TSET BAUD	Set the serial port's communication parameters for a device. **Note:** *This value is not saved in Non-Volatile memory, and is reset to default (9600,8,N,1) at power-up.* Syntax: `TSET BAUD <baud>,<parity>,<data>,<stop> [<Enable | Disable>]` Variables: • Baud: 115200, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 150. • Parity: N (none), O (odd), E (even), M (mark), S (space) • Data: 7 or 8 data bits • Stop: 1 or 2 stop bits Example: `SEND_COMMAND ENZO_RS232,"'TSET BAUD 115200,N,8,1'"` Sets the ENZO_RS232 port's communication parameters to 115,200 baud, no parity, 8 data bits, 1 stop bit.
TV.CHANNEL	Changes the current channel playing in the TV app. This command will also launch the TV app if it is not already running. **AUTOSTART** Syntax: `TV.CHANNEL-<channel id>/<channel name>` Variables: *channel id* = specifies which channel number to change to. *channel name* = specifies which channel name to change to.
?TV.CHANNEL	Returns the current persisted channel which is the last channel set. If the media player is streaming then it is the current persisted channel. Syntax: `?TV.CHANNEL` Example Response: `Line 1 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-id,12]` `Line 2 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-name,PS-TV2]` `Line 3 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-description,PS-TV2]` `Line 4 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-icon-0,http://localhost/files/media/images/TV/channel_icon.png]` `Line 5 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-icon-1,http://localhost/files/media/images/TV/channel_icon.png]` `Line 6 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-icon-2,http://localhost/files/media/images/TV/channel_icon.png]` `Line 7 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-level,0]` `Line 8 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-serviceId,]` `Line 9 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-type,v2Liveh264]` `Line 10 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-unmanaged,yes]` `Line 11 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-mimetype,video/mpg]` `Line 12 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-uri,videocastmcast://239.35.84.84:5500/]` `Line 13 (11:56:17):: Command From [6001:1:1]-[TV.CHANNEL_INFO-used_uri,udp://239.35.84.84:5500/]`
?TV.CHANNEL_INFO	Lists Meta Data for a Channel. Use ?TV.CHANNELS to obtain the list of channel names and ids to choose from. **Note:** *This command will not return a response with valid information unless the TV app is launched first or the channel list is set using TV.CHANNELS.* Syntax: `?TV.CHANNEL_INFO-<channel name>/<channel id>` Variables: *channel name* = specifies which channel name to request information from. *channel id* = specifies which channel number to request information from. Example Response: Example Response is the same as ?TV.CHANNEL above.

Continued ↴

NetLinx Commands	Description
TV.CHANNELS	When Vision 2 is disabled this command specifies the location of the TV channel list to use for programming and whether to keep the list in following sessions. When Vision 2 is enabled this command specifies the Vision 2 server. Syntax: `TV.CHANNELS-"url",<retain>` Variables: *url* - the location of the TV channel list file, Vision2 server, or url channel list configuration. *retain* - true to save configuration when exiting session or false for temporary channels **Note:** *If a retain value is not provided then it defaults to "false".* Examples: `TV.CHANNELS-ftp://username:passwordAsNumber@ip/file.json,true` `TV.CHANNELS-http://10.35.84.138,true` ***CAUTION:*** *The Vision 2 server must be enabled prior to issuing this command.*
?TV.CHANNELS	This command will list the available TV channels. **Note:** *This command will not return a response with valid information unless the TV app is launched first or the channel list is set using TV.CHANNELS.* Syntax: `?TV.CHANNELS` Example Response: `Line 57 (15:22:58):: Command From [10021:1:1]-[TV.CHANNEL-TWB-TV1,1]` `Line 58 (15:22:58):: Command From [10021:1:1]-[TV.CHANNEL-PS-TV1,10]` `Line 59 (15:22:58):: Command From [10021:1:1]-[TV.CHANNEL-PS-TV2,11]` `Line 60 (15:22:58):: Command From [10021:1:1]-[TV.CHANNEL-SWB1-1-TV1, 20150120120411086]` `Line 61 (15:22:58):: Command From [10021:1:1]-[TV.CHANNEL-SWB1-2-TV2, 20150120120433092`
TV.CONFIGURATION	This commands sets the video source to either Vision 2 or a TV configuration URL. Syntax: `TV.CONFIGURATION-<true/false>` Variables: false = Sets Vision2 false and loads Enzo TV configuration URL true = Enables the Vision2 configuration. **Note:** *You still need to set the URL/FILE with TV.CHANNELS after setting the TV.CONFIGURATION for the first time.*
?TV.CONFIGURATION	Queries the Enzo TV Configuration and returns a value of true for Vision2 or false for file or URL channel list configuration. Variables: true = Vision2 false = file or url channel list configuration.
TV.CONTROLLER	Changes the channel like a remote control would. This only includes live channels. For example, if you are playing a VOD and call TV.CONTROLLER-UP then you will change up one channel from the last Live Channel. Syntax: `TV.CONTROLLER-<channel>` `Variables:` UP - change up one channel from the last Live Channel DOWN - change down one channel from the last Live Channel PREVIOUS - return to the previous Live Channel Example: `TV.CONTROLLER-UP` **Note:** *The UP and DOWN in the list of live channel and does not increment for up and decrement for down. So, if you are making it like some remotes you will need to reverse it.*

Continued ↴

NetLinx Commands	Description
TV.NAVIGATION	Used to set the default action when the TV app is launched and to persist the location shown when Enzo TV is loaded. `TV.NAVIGATION-<location>,<retain>` Variables: *locations* = guide, video, or vod (Video On Demand) • TV.NAVIGATION-guide – opens the guide to the live channel listings • TV.NAVIGATION-video – closes the guide if it is already opened • TV.NAVIGATION-vod – opens the guide to the VOD listings *retain* = is true/false and determines if the location setting is saved for when Enzo TV is loaded. • TV.NAVIGATION-guide,true – sets the default of the navigation startup. With guide,true when the TV app is launched for the first time the guide will be shown • TV.NAVIGATION-video,true – Same setting as above but with video,true when the TV app is launched for the first time video only will be shown, not the guide. • TV.NAVIGATION-vod,true – This setting would be invalid. We do not allow the VOD guide to be set as the startup navigation option so if this command was sent nothing would be changed. **Note:** *The Navigation Startup setting only applies when the TV application is first launched during an Enzo session. If a user returns to the TV application during a session, the guide may not be shown. If this occurs, the user can press the Guide button in the upper right corner of the screen or press the Enter key.*
TV.VOD	This command starts a video as a VOD with the TV app. Syntax: `SEND_COMMAND <DEV>, "'TV.VOD-<url>'{,<name>}'"` Example: `TV.VOD-http://10.35.84.138/v2/Archives/Archive1/0005/09/TS/High/video.ts,Bugs`
UP	Issues an ARROW-UP keystroke to the active app. Actual results will be app-specific. Syntax: `SEND_COMMAND <DEV>, "'UP'"`
VIEW	Launches a video stream on Enzo. Value between "{ }" is optional. **AUTOSTART** Syntax: `SEND_COMMAND <DEV>, "'VIEW-<mimetype>,<url>{,<channel name>}'"` Example: `VIEW-video/mpg,udp://234.5.0.1:5500` `VIEW-video/mpg,udp://234.5.0.1:5500,DIY Network` Related Command: See VIEW.CLOSE To stop an active stream Syntax: `SEND_COMMAND <DEV>, "'VIEW.CLOSE'"` Example: `VIEW.CLOSE`
?VIEW	Queries the current video if any. Returns the following commands when a video is in use: `VIEW.RESPONSE` `VIEW.STATE-media state` `VIEW.URL-url` `VIEW.CHANNEL-channel name` `VIEW.MIME-mime type` Syntax: `SEND_COMMAND <DEV>, "?VIEW"`
VIEW.CLOSE	Stops the media and closes the Media Player. Syntax: `SEND_COMMAND <DEV>, "VIEW.CLOSE"`

Continued ↴

NetLinx Commands	Description
VIEW.MEDIA_CONTROLLER	The VIEW.MEDIA_CONTROLLLER–**xxxx** command has multiple qualifiers depending on which task is being performed. Refer to the following command qualifiers: Where **xxxx** = • *hide* - Enzo will immediately hide the media controller. • *show,time* - Enzo will immediately show the media controller for the amount of time requested by the command unless the stream state is changed from user input or stream outage. Time is in seconds. A negative value will show the media controller indefinitely until the local user dismisses it or a new video is started. **Note:** *This command will be ignored if the channel guide is currently displayed when this command is received. First close the channel guide using the command TV.NAVIGATION* on page 29 *(i.e., TV.NAVIGATION-video).* • *lock* - Enzo will lock out user/device input from show/hide the media controller. Only commands will show/hide the media controller. • *unlock* - Enzo will unlock show/hide for the media controller so that the local user/device can control the media controller. This is the default setting. Lock and Unlock only impact the current stream/media controller instance. As such you will need to lock when starting each stream. Syntax: `SEND_COMMAND <DEV>, "VIEW.MEDIA_CONTROLLER-<command>{,<time in seconds>"}` **Note:** *No space after comma for time.*
VOLUME	Sets the volume of the attached display. Arguments: *volume* - the new volume (0-255) Syntax: `SEND_COMMAND <dev>,'VOLUME-<volume>'` Example: `SEND_COMMAND 1001:1:0, 'VOLUME-128'`
VOLUME.MUTE	Mutes the audio on the attached display. Arguments: **mute** - • *true/on/1* = muted • *false/off/0* = unmuted Syntax: `SEND_COMMAND <dev>,'VOLUME.MUTE-<mute>'` Example `SEND_COMMAND 1001:1:0,'VOLUME.MUTE-off'`
WAKE	Wakes the Enzo display Syntax: `SEND_COMMAND <dev>,'WAKE'`
WEB	Opens a web page. **AUTOSTART** Syntax: `SEND_COMMAND <DEV>, "'WEB-<uri>>'"` `SEND_COMMAND <DEV>, "'WEB-<uri>,<title>'"` `SEND_COMMAND <DEV>, "'WEB-<uri>,<title>,<mobile: true|false>'"` Variables: url = The URL to display title = The title of the display. This argument is optional. mobile = true|false: 'True' will request the URL to load the mobile version of the web page. 'False' will request the URL to load the desktop version of the web page. This argument is optional. **NOTE:** *If this option is omitted, it defaults to 'false' thus resulting in a 'desktop' version of the requested URL.* Example: `SEND_COMMAND <dev>,'WEB-http://www.amx.com,My favorite web site,false'` **Note:** *Title suggested length is 120 characters or less. However, no hard limit is set.*
WEB.CLOSE	Closes an open web page. Syntax: `SEND_COMMAND <DEV>, "'WEB.CLOSE'"`

Continued ↴

NetLinx Commands	Description
WEBU	This command initiates a web based update of the Enzo firmware from a specified location and kit. Syntax: `WEBU,<url>,<reboot>` Variables: URL - The location of the kit file for the Web Update. Reboot - True to reboot after the update or False to not reboot after the update. Example: `WEBU, http://10.35.94.25:80/SW3211-01_Enzo_v1_5_15-Full.kit,REBOOT=TRUE` **IMPORTANT:** *This is the only non standard command with a comma for a command delimiter.*

Enzo System Responses

The following table lists status events generated by Enzo, some generated as command responses and some as unsolicited responses.

Event	Description
SESSION	Asynchronous event A new session is started `'SESSION-TRUE'` A session is ended `'SESSION-FALSE'`
Possible states sent to the NetLinx programmer for Streaming Content:	
ACTIVITY.STATE	Automatically sent out when apps are started. ACTIVITY.STATE-ONSTARTING,<simple app name> Variables" simple app name = Home, Browser or MirrorOp **IMPORTANT:** *Important: Only a few applications have a simple application name. So don't expect this response for all activities.* Example Responses: `ACTIVITY.STATE-ONSTARTING,HOME` `ACTIVITY.STATE-ONSTARTING,BROWSER`
ALERT.CLOSED	If using "AlertActivity" a response is sent when the alert closes. It will include yes, no, ok, and command base on how it is closed. Example responses: `ALERT.CLOSED-yes` `ALERT.CLOSED-no` `ALERT.CLOSED-ok`
ERROR	Sent when the URL is invalid, or command is so malformed or that it never gets sent. Example responses: `ERROR-Invalid Command is sent when the URL is invalid, or command is so malformed or that it never gets sent.` `ERROR-Exception encountered is sent when the processing of a command results in an exception`
MIRROROP.CONNECTION	`MIRROROP.CONNECTION` `MIRROROP.CONNECTION-NEW` `MIRROROP.CONNECTION-LOST`
MIRROROP.CONNECTIONS	Sent when a new MirrorOp presenter is connected to Enzo. Syntax: `MIRROROP.CONNECTION-<senderId>,<login name>,display:<display>:<fullscreen/quadrant>,` Variables: *login name* - String set by the sender upon login. *display* - is the current display being shown 0-128. Example: (response for sender with senderId of 26) `MIRROROP.CONNECTIONS` `MIRROROP.CONNECTION-26,username,display: 0:quadrant: 2 (Top Right)]`
NOT_FOUND	Is sent when a stream does not start or when it has not streamed for 20 seconds. Example response: `VIEW.STATE-NOT_FOUND`
NOT_STREAMING	Is sent when a stream is interrupted but still possibly good. Example response: `VIEW.STATE-NOT_STREAMING`

Continued ↴

Event	Description
STREAMING	Sent when a stream begins streaming at the beginning or starting from a network interruption. Example response: `VIEW.STATE-STREAMING`
The following unsolicited responses will be returned as COMMANDS.	
CONTENT.SOURCE.CHANGED	This event will be delivered to the NetLinx program anytime the content source used in the API cursor is changed. Example: `CONTENT.SOURCE.CHANGED-<source-id>,<source-name>, <source-root-path>,<source-is-ready:true|false>` Content source changes include the following events: • The content source is changed/updated via the content API commands. `(CONTENT.SOURCE-<source-id>)` • A removable content source is attached or removed. • An authenticated content source is authenticated/unauthenticated.
CONTENT.PATH	This event will be delivered to the NetLinx program anytime the current content path on the API cursor is changed. Example: `CONTENT.PATH-<source-id>,<path>` Content path changes include the following events: • The content path is changed/updated via the content API commands. `(CONTENT.PATH-<path>)` A content source change occurs when: • The content source is changed/updated via the content API commands. `(CONTENT.SOURCE-<source-id>)` • A removable content source is attached or removed. • An authenticated content source is authenticated/unauthenticated.

Programming a Channel List

The following example is provided to help programmers set up a simple channel list using just the required name and url.

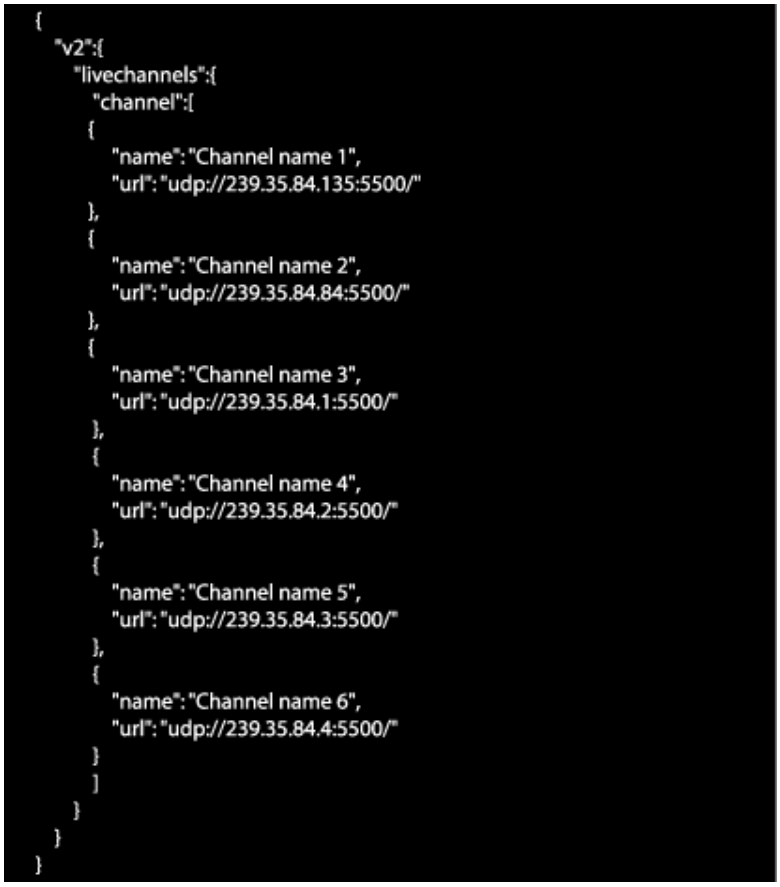


FIG. 17 ***Example of a Simple JSON Program Channel List***

Note: *For each channel, the administrator must provide a name and url.*

Each channel may optionally include id, description, and/or icon as shown below for each channel:

- Id - a unique identifier for a given channel. If 1 channel id is provided, then you should supply a unique id for all channels.
- Description - describes what is on a channel.
- Icon - is a URL to a custom image for a channel.

more ↴

Note: *Icons are only shown if enabled from the Web UI.*

```
{
  "v2":{
    "livechannels":{
      "channel":[
      {
        "name":"Channel name 1",
        "id":"1",
        "url":"udp://239.35.84.135:5500/",
        "description":"channel description goes here 1",
        "icon":"http://10.35.94.29/files/media/images/TV/icon_icon_enzo_tv.png"
      },
      {
        "name":"Channel name 2",
        "id":"2",
        "url":"udp://239.35.84.84:5500/",
        "description":"channel description  goes here 2",
        "icon":"http://10.35.94.29/files/media/images/TV/icon_icon_enzo_tv.png"
      },
      {
        "name":"Channel name 3",
        "id":"3",
        "url":"udp://239.35.84.1:5500/",
        "description":"channel description goes here 3",
        "icon":"http://10.35.94.29/files/media/images/TV/icon_icon_enzo_tv.png"
      },
      {
        "name":"Channel name 4",
        "id":"4",
        "url":"udp://239.35.84.2:5500/",
        "description":"channel description goes here 4",
        "icon":"http://10.35.94.29/files/media/images/TV/icon_icon_enzo_tv.png"
      },
      {
        "name":"Channel name 5",
        "id":"5",
        "url":"udp://239.35.84.3:5500/",
        "description":"channel description goes here 5",
        "icon":"http://10.35.94.29/files/media/images/TV/icon_icon_enzo_tv.png"
      },
      {
        "name":"Channel name 6",
        "id":"6",
        "url":"udp ://239.35.84.4:5500/",
        "description":"channel description goes here 6",
        "icon":"http://10.35.94.29/files/media/images/TV/icon_icon_enzo_tv.png"
      }
      ]
    }
  }
```


FIG. 18 ***Example of a Detailed JSON Program Channel List***

More can be found out about the JSON (JavaScript Object Notation) format at: http://www.json.org/

There are also example files available for download from the product page located under the "Device Modules" section on the right side of the page:

```
http://www.amx.com/products/NMX-MM-1000.asp
```

Once your configuration file has been built, it can be uploaded to Enzo using the NetLinx command:

```
TV.CHANNELS-"url"<retain>
```

This command specifies the TV channel list location to use for programming and whether to save the list when exiting session.

Syntax:

```
TV.CHANNELS-"url"<retain>
```

Variables:

url - the location of the TV channel list file or url channel list configuration.

retain - true to save configuration when exiting session or false for temporary channels

Example:

```
TV.CHANNELS-ftp://username:passwordAsNumber@ip/file.json,<retain>
```

Another way to upload this file is using the Enzo WebUI as described in the Enzo Administrators Guide available at http://www.amx.com/products/NMX-MM-1000.

Enzo Keypad

The Enzo keypad provides a simple web-based user interface for NetLinx control systems which can be used to control Enzo. The Enzo keypad can be programmed and the settings uploaded to a NetLinx Master using AMX's NetLinx Studio application. The Enzo keypad works in an identical fashion to a physical keypad. FIG. 19 displays an example of the Enzo keypad.

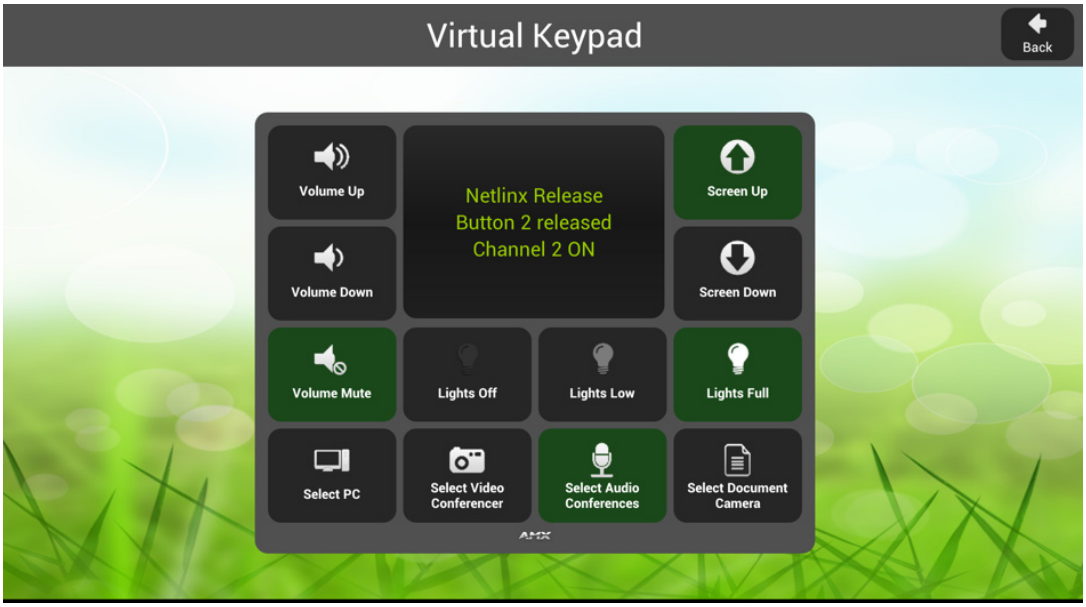


FIG. 19 Enzo Keypad (example)

Installing the Enzo Keypad onto a NetLinx Master

A source code example for the Enzo keypad is available for download at www.amx.com. The example code is stored in a .zip file. The .zip file contains two files:

- EnzoKeypad.axi - An .axi file is an include file that is included in the compiling process without having to reside in the Master Code file itself. Include files are effectively linked to the Source Code file, and must be called in the Master Source Code.
- EnzoKeypadExample.axs - The source code file containing code that is sent to the NetLinx Master.

The example code serves as a starting point for creating a virtual keypad. The code can be customized to implement functionality for button events, line text feedback, and button label text.

Perform these steps to install the Enzo keypad onto a NetLinx Master.

NOTE: *Before starting, verify that the latest version of NetLinx Studio 3 (available via free download at www.amx.com) is installed.*

1. Add the EnzoKeypadExample.axs file to the workspace in NetLinx Studio. Right-click the Source folder in the Workspace bar of NetLinx Studio, select **Add Existing Source File**, and browse for the .axs file to add it to the workspace.
2. Add the EnzoKeypad.axi file to the workspace in NetLinx Studio. Right-click the Include folder in the Workspace bar of NetLinx Studio, select **Add Existing Include File**, and browse for the .axi file to add it to the workspace.
3. In the application workspace, add the custom code to implement functionality for button events, line text feedback, and button label text. This includes constant definitions, variable definitions (for ramping), module definitions, data events, and button events.
4. Ensure the virtual device definition for the Enzo Keypad has a unique device number, as there may be other previously-defined virtual devices.
5. Click the **Build Active System** button to build the system and ensure all code compiles without errors.
6. Select the NetLinx Master to which to transfer the code.
7. Click **Send** to transfer the code to the NetLinx Master.

3000 RESEARCH DRIVE, RICHARDSON, TX 75082
AMX.com | 800.222.0193 | 469.624.8000 | +1.469.624.7400 | fax 469.624.7153

Last Revised:
10/04/2016