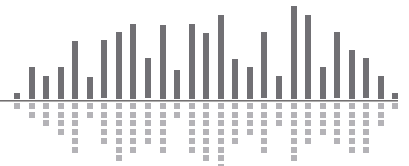
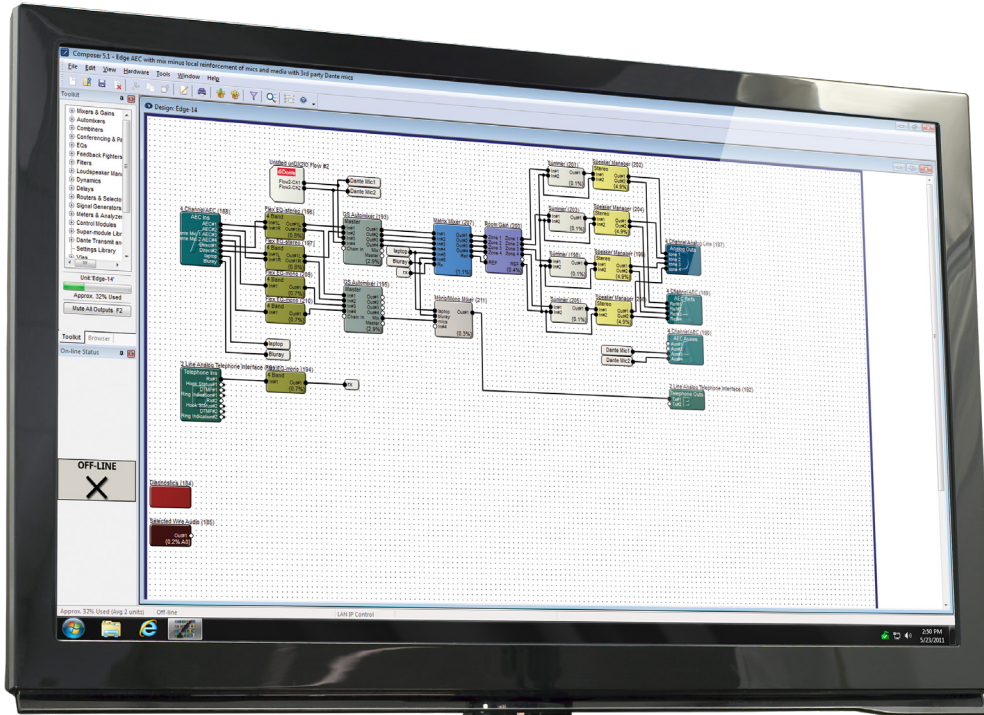


Composer v7.0 Control Protocol



Contents

Introduction

About This Document	3
Conventions Used In This Document	3

General Notes

Connections	3-4
RS-232 Port Configuration	3
Ethernet Port Configuration	3
Ethernet Control	3
RS-485 Control	4
Site Identifier	4

Parameter Notes

Faders	5
Faders with User Min and User Max Limits	5
Buttons	6
Input and Output Selectors	6
Meters	6
LEDs	6
Other Parameters	6
Diagnostic Controls	6
Parameters That Cannot Be Controlled Externally	7

Getting Started

Protocol	7
----------	---

Control Commands

(CS) Controller Set	7-13
(CSQ) Controller Set Quick	7
(CC) Change Controller	8
(GS) Get Controller	8
(GS2) Get Controller with Controller Number	8
(GPR) Get Preset	9
(GSB) Get Controller Block	9
(GSB2) Get Controller Block with Controller Number	9
(GSB3) Get Controller Block with Controller Number	10
(LP) Load Preset	10
(FU) Flash Unit LEDs	11
(SSYSS) Set System String	11
Dialing VoIP Strings	11
(GSYSS) Get System String	12
Pushing GSYSS String Values	12
	13

Push Commands

13-15

Commands Related to Push

(PU) Global Push Enable/Disable	15-18
(PUE) Push Enable	15
(PUD) Push Disable	16
(GPU) Get Push-enabled Controllers	16
(PUR) Push Refresh	17
(PUC) Push Clear	17
(PUI) Set Push Interval	18
(PUT) Set Push Threshold	18

Setup Commands

(SB) Set Baud	19
(SQ) Set Quiet Mode	19
(EH) Set Echo Mode	19

System Commands

(NOP) No operation	20
(Q!) Quit TCP/IP session	20
(RI) IP address	20
(V) Version	20
(R!) Reboot	20

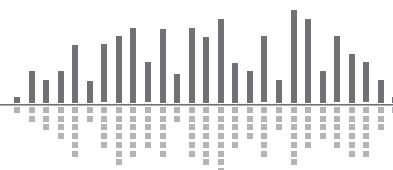
Special Command Prefixes

(\$e) Echo Mode	21
(\$v) Verbose Mode	21
(\$q) Quiet Mode	21

Super Matrix Mixer Control Protocol

21-24

Background	21
Features to Control	21
Commands	21
Basic Examples	22
Range Examples	23
Return Value	23
Polling for Changes	24



Introduction

About This Document

The purpose of this document is to introduce the Symetrix Control Protocol for Composer. This document defines and illustrates the protocol used to communicate with Edge, Radius, Radius AEC, xControl, or other compatible Symetrix units via a third-party control interface using RS-232, TCP/IP, or UDP/IP.

Compatible units can be controlled by third-party controllers such as AMX or Crestron models, or any RS-232 or Ethernet equipped unit that can be adapted to the Composer control protocol. The protocol consists of human-readable text commands and responses.

Control is achieved by using a scheme of user-assigned controller numbers. Nearly anything that can be adjusted from the Composer control and configuration application can be controlled externally by referencing the appropriate controller numbers that are defined using Composer.

Conventions Used In This Document

Values enclosed in [square brackets] are optional parameters and do not need to be included. If omitted, default values will be used as described for each command.

The term “control application” is used to refer to Composer software provided by Symetrix to configure and control Symetrix units.

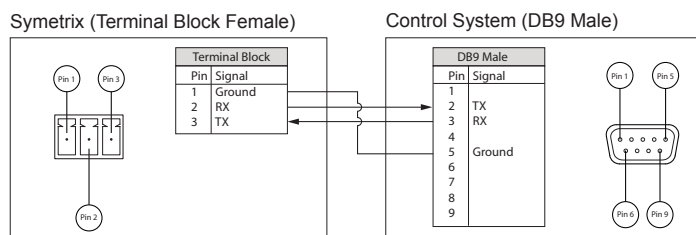
General Notes

Connections

Units are equipped with an Ethernet port and one or more RS-232 ports. Either port may be used to control the unit. Ethernet is generally preferred because of the simplified set-up and higher data rates.

RS-232 Port Configuration

Connect your RS-232 based accessory remote to the RS-232 port. For testing purposes, a terminal emulation program such as HyperTerminal, TeraTerm, or puTTY can be run on a PC instead. Typically this connection requires a “straight through” cable, but a null-modem cable may be required depending on the manufacturer. In general, if the genders of the two connectors you are trying to connect are the same, a “null modem” cable is required. Set up the controller for baud: 57600, data bits: 8, stop bits: 1, parity: NONE. No handshaking or flow control is used. The straight-through pin out for a typical application is shown in the following figure.



The commands **Set Baud**, **Set Quiet Mode** and **Set Echo** affect the RS-232 port. Composer determines the start-up values of these settings, though this can be changed during a session. The unit's default settings (Baud 57600, Quiet Mode ON, Echo OFF) are typical for most applications. These commands are documented for reference on page 18.

Ethernet Port Configuration

Generally, no special configuration is required for the Ethernet port. The Ethernet port on the unit may be used for both the Composer control application and for external control.

Take note of the unit's IP address (listed in the System Manager), as you will need to send all commands to this address.

The commands **Set Quiet Mode** and **Set Echo** affect the Ethernet port. Composer determines the start-up values of these settings, though this can be changed during a session. The unit's default settings (Quiet Mode ON, Echo OFF) are typical for most applications. These commands are documented for reference on page 18.

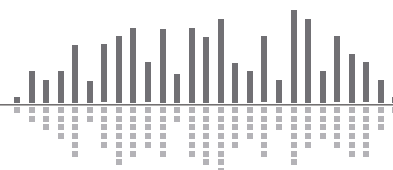
Ethernet Control

The Ethernet protocol uses the same RS-232 command protocol over an Ethernet network. Both TCP/IP and UDP can be used to control the system. TCP/IP control functionality was added in Composer 3.0. TCP/IP control does not use any of the options or escape sequences found in Telnet. Up to six TCP/IP telnet sessions can be active at one time (four with versions prior to Composer 6.0).

If a seventh TCP/IP connection is initiated, the least recently used session will be automatically closed. Sessions may also be closed manually with the **Q!** command described later in this document.

Control systems should avoid closing and re-opening TCP connections if possible. Keeping a single TCP session open to send multiple commands through will result in much better performance than opening and closing a session for each command.

To control the system with Ethernet, command strings are sent as the payload of a UDP/IP or a TCP packet. The following rules should be observed in sending commands:



- Commands should be sent to UDP or TCP port number 48631 of the Edge or Radius unit's IP address. The IP address may be found using the Connection Wizard or on some units' front panel displays.
- Commands should be formatted exactly as defined in this document and include a carriage return that terminates the command.
- Command strings may or may not include a zero termination character.
- Commands should not be broken up across multiple packets.
- If high reliability communications are required, responses to commands should be analyzed for success.

Responses to commands will exhibit the following behavior:

- Responses to each command issued are returned in a single packet unless the response is larger than a single packet can hold. Responses will not have any single carriage return-terminated line broken up across packets unless there is no carriage return in the data of the response.
- Responses are returned only to the IP address and source port number that sent the packet. Other connections will see responses if the particular control numbers are 'push' enabled.
- Responses follow the configuration of the port, just as if it were an RS-232 port. For example, echo mode and quiet mode are kept independently for the RS-232 port, the UDP port and the TCP port.
- Responses do not include a zero-termination character.
- All transmissions originating from units will either be responses to commands or pushed data.

Each command sent to a Symetrix unit contains information in the Ethernet packet header as to who sent the command, and hence, where a response will be sent. This source information is saved when a packet is received by a Symetrix unit. For UDP, all responses go to the last received IP address and port and this IP address and port number are saved in non-volatile memory across power cycles. For TCP, the control system must re-initiate the TCP connection after a power cycle.

Until the first command is received, responses will not know where they are supposed to be sent. This normally is not an issue as communication from the Symetrix unit is generally a response to a command. However, if the Symetrix unit is set up to push control data, it will also be pushed out the TCP and UDP ports. If no valid packets have ever been received by a Symetrix unit and no TCP/IP connections are active, pushed data will not be sent out the Ethernet port. If a Symetrix unit and a TCP connection is active, then pushed data will be transmitted over TCP. If valid packets have been received over UDP by a Symetrix unit, then pushed data will be transmitted over UDP.

Pushing of data can be controlled independently for RS-232 and Ethernet ports.

Note: The RS-232 serial port and the Ethernet port are essentially independent. They maintain separate settings for quiet and echo modes. Commands sent to one port are not echoed out the other, and responses are sent only to the port from which the command was received. Hence, the two ports will not necessarily send out the same data. One exception to this is push data and strings, which are sent out both RS-232 and Ethernet ports in parallel.

RS-485 Control

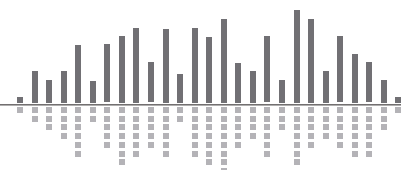
RS-485 control is generally done using one or more of the Symetrix ARC (Adaptive Remote Controller) units. Further discussion of RS-485 and the ARCs can be found on the Symetrix web site.

Site Identifier

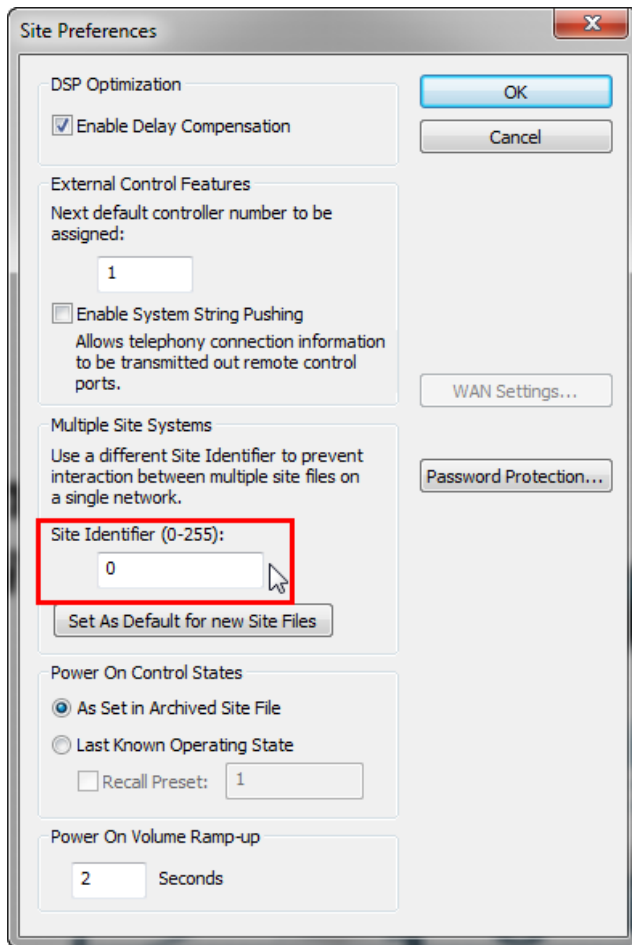
By default, any control data sent to a Symetrix unit (via ARC, RS-232, UDP/IP, or TCP/IP) will be re-broadcast over Ethernet by the unit that receives the command to all units that are connected via Ethernet. This ensures that multiple Symetrix units with the same Site Identifier that are using the same control numbers will also receive and interpret the command and respond to the command appropriately (i.e., adjust faders, button states, etc.).

When the system consists of multiple independent site files on the same network and you do not want control data for common control numbers to be shared between these different site files, each site file must have a unique site identifier set in Composer. Unique Site Identifiers in the site file inform other systems with different site identifiers on the same network to ignore the re-broadcasted commands.

Using unique Site Identifiers for different site files on the same network does not require any change in how the systems are controlled from a third-party control system. The third-party control system continues to send commands to the Symetrix device to be controlled and that device re-broadcasts that command along with the additional information of the site identifier onto the network. Symetrix units with Site Identifiers that are different from the site identifier in the re-broadcasted command will simply ignore the re-broadcasted command.



To set the identifier, within Composer, navigate to Tools > Site Preferences and change the site identifier as shown in the following figure.



Parameter Notes

Faders

Faders can be controlled to the limits of their minimum and maximum values shown in the Design View screens in Composer. A controller position of zero (0) will cause the minimum fader position to be realized. A controller position of 65535 will cause the maximum fader position to be realized. Increasing positions will move the fader linearly in dB.

Most volume faders have a range of -72dB to +12dB. In these cases, the following formula can be used to convert from controller position to dB:

$$\text{Volume dB} = -72 + 84 * (\text{CONTROLLER POSITION} / 65535)$$

If CONTROLLER POSITION = 0, Volume dB = OFF

For these faders, a value of 780 is approximately equal to a 1dB change in gain.

Note that some faders have a different range than -72 to +12dB. In this case, the formula will depend upon the actual fader range. The more general formula is shown below:

$$\text{Volume dB} = \text{MINIMUM VALUE} + (\text{MAXIMUM VALUE} - \text{MINIMUM VALUE}) * (\text{CONTROLLER POSITION} / 65535)$$

Where MINIMUM VALUE is the fader's lower limit in dB and MAXIMUM VALUE is the fader's upper limit in dB.

Faders with User Min and User Max Limits

There are some applications where it is desired to limit the range of adjustments for an end user, perhaps to allow a more useable range of adjustment than the full range of the fader or to prevent the system from getting too loud. To limit the gain range of a fader to a user minimum and user maximum range, the following formulae may be used.

In the following discussion, it is assumed that the range of the variables (in dB) are as follows:

$$\text{MAXIMUM VALUE} \geq \text{User Maximum} \geq \text{User Volume} \geq \text{User Minimum} \geq \text{MINIMUM VALUE}$$

The controller position is calculated the same as described previously:

To determine the controller position based on a desired User Volume in dB, then the controller value is calculated as follows:

$$\text{Controller Position} = (\text{User Volume in dB} - \text{MINIMUM VALUE}) * 65535 / (\text{MAXIMUM VALUE} - \text{MINIMUM VALUE})$$

The percentage value of the fader control relative to the User Min and User Max values is computed as:

$$\text{Volume Percent} = (\text{User Volume in dB} - \text{User Min}) / (\text{User Max} - \text{User Min})$$

The controller position required to set the fader to a percentage of the user min and user max may be computed as follows:

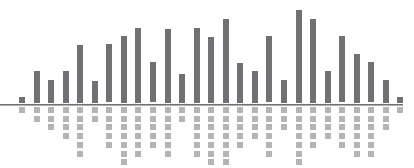
$$\text{Controller Position} = (\text{Volume Percent} * (\text{User Max} - \text{User Min}) + (\text{User Min} - \text{MINIMUM VALUE})) * 65535 / (\text{MAXIMUM VALUE} - \text{MINIMUM VALUE})$$

For example, if the Maximum and Minimum Values are +12 and -72, respectively, and the desired User Max and User Min are +6 and -30 respectively, then the controller value for setting the fader to -8dB is:

$$\text{Controller Position} = ((-8) - (-72)) * 65535 / ((+12) - (-72)) = 49931$$

The controller position corresponding to 35% volume with the same User Max and User Min range would be:

$$\text{Controller Position} = (35\% * ((+6) - (-30)) + ((-30) - (-72))) * 65535 / ((+12) - (-72)) = 42598$$



Calculating the Volume in dB from the Volume Percentage would be done as follows:

$$\text{Volume in dB} = (\text{Volume Percent} * (\text{User Max} - \text{User Min})) + \text{User Min}$$

In this example, the volume in dB corresponding to 50% of the volume would be calculated as:

$$\text{Volume in dB} = 50\% * ((+6) - (-30)) + -30 = -12 \text{ dB}$$

Buttons

Buttons such as a mute or bypass can be controlled similarly with controller positions by sending the minimum value (0) to turn the switch off (button not pushed) and sending the maximum value (65535) to turn the switch on (button pushed).

Input and Output Selectors

A value of zero (0) will select the first input or output and a value of (65535) will select the last input or output. Other values are selected by sending evenly spaced (linear) values as shown by the formula below:

$$\text{Controller Value} = (\text{INPUT NUMBER} - 1) * 65535 / (\text{NUMBER OF INPUTS} - 1)$$

Note: for an output selector, replace “INPUT” with “OUTPUT” above.

Meters

Meters can be read via RS-232 or Ethernet. The read back value will be linear in dB with 65535 representing +24 dBu (0 dBFS) and 0 representing -48 dBu (-72 dBFS) (or less). The formula below can be used to calculate a dB reading from a controller value:

$$\text{Level dBu} = 72 * (\text{CONTROLLER VALUE} / 65535) - 48$$

If CONTROLLER VALUE = 0, Level dBu <= -48 dBu

Input and output meters in some other modules such as Compressors and AGCs can also be read. In this case, the read back value will be linear in dB with 65535 representing the maximum value shown on the meter and 0 representing the minimum value shown on the meter (or less). The formula below can be used to calculate a dB reading from a controller value:

$$\text{Level dB} = (\text{MAXIMUM VALUE} - \text{MINIMUM VALUE}) * (\text{CONTROLLER VALUE} / 65535) + \text{MINIMUM VALUE}$$

$$\text{If CONTROLLER VALUE} = 0, \text{Level dB} \leq \text{MINIMUM VALUE}$$

Where **MINIMUM VALUE** is the meter’s lower limit in dB and **MAXIMUM VALUE** is the meter’s upper limit in dB.

Control meters in dedicated control meter modules can also be read. In this case, the read back value will be linear

with 65535 representing 100% and 0 representing 0%. The formula below can be used to calculate a percentage reading from a controller value:

$$\text{Level \%} = 100 * (\text{CONTROLLER VALUE} / 65535)$$

Note: Meters are a “read-only” parameter. Attempting to change the meter value will have no effect.

LEDs

LEDs in various modules can be read via the control protocol. The read back value will be either 65535 if the LED is on or 0 if the LED is off.

Note: LEDs are a “read-only” parameter. Attempting to change the LED value will have no effect.

Diagnostic Controls

Connected LED

The read back values are:

0 = Disconnected

65535 = Connected

Network Diagnostics Panel

DHCP Enabled LED - Only DHCP Enable can be read unless using SymVue. The read back values are:

0 = DHCP Disabled

65535 = DHCP Enabled

Power Supply Diagnostics Panel

All voltages are in .001V increments offset by 32768. The read back values of 0-65535 translate to -32.768V-32.767V.

Status LED - The read back values are:

0 = OK

65535 = Failed

Thermal Diagnostics Panel

Peak temperature date and time can only be read in SymVue. All temperatures are in .01 degree C increments offset by 32768. The read back values of 0-65535 translate to -327.68V-327.67V. The Fan speed is 0-65535 Hz.

Status LED – Send -65535 to clear, The read back values are:

0 = OK

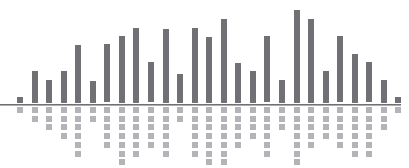
65535 = Failed

Control Diagnostics Panel

All read back values:

0 = Inactive/Off

65535 = Active/On



Other Parameters

Many other parameters such as compression ratios, delay times, EQ settings and pans can also be controlled externally. In fact, almost any DSP parameter in Composer can be controlled. For other parameter types, as in the above examples, sending a value of zero (0) will set the parameter to its minimum value and sending a value of (65535) will set it to its maximum value. Ratios, frequencies, width/Q, and attack/release/hold times all use a logarithmic scale. Pans and delay times use a linear scale. Quantities expressed in dB such as gains, volumes, thresholds and depths are linear in dB. When in doubt, experiment by changing a value from the control application and reading it back via RS-232 or Ethernet.

Parameters That Cannot Be Controlled Externally

The following Composer features cannot be controlled externally. In some cases, work arounds are suggested.

	FUNCTION	DESCRIPTION
1.	Oscilloscope:	Trigger and Auto-Range controls
2.	Parametric/British EQ:	Flatten All and Octaves/Q buttons
3.	Graphic EQ:	Flat button
4.	Feedback Fighters:	Response Preset buttons
5.	Matrix:	Mute All and Disconnect All buttons (use individual mute or disconnect buttons instead)
6.	Mixers:	Link buttons (assign same controller to all linked faders instead)
7.	SPL Computers:	Calibrate button
8.	Loudspeaker Managers:	Flatten All
9.	Send/Return:	Meters (use meters inside I/O blocks)
10.	Delay:	Default temperature button
11.	Meter Bar:	All meters
12.	Fader Control Input:	Log/linear buttons
13.	String Output:	All parameters
14.	RS-232/485 Input:	Controller number

Getting Started

Protocol

The Control Protocol is a text-based (ASCII string) protocol. Commands are sent with simple character strings with terms separated by spaces and completed with a carriage return character <CR> (ASCII code decimal 13). The general form for commands is:

<COMMAND> <PARAMETER1> <PARAMETER2> ... <CR>

A white space character (space or tab) must be included between the command and each parameter. Extra white space characters can be sent for readability if desired. In this document a single space will be used. If a command is accepted, the unit will respond to each command with an acknowledgement string whose syntax varies with each command.

Control Commands

(CS) Controller Set

Use this command to move a controller position to a new absolute value. The command must specify the controller number and the new controller position. The syntax of the command is:

CS <CONTROLLER NUMBER> <CONTROLLER POSITION><CR>

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer, and <CONTROLLER POSITION> is a 16-bit number in decimal (0-65535).

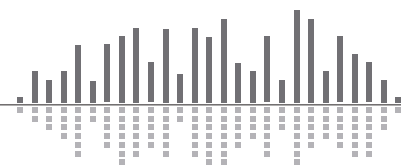
If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the specified controller number does not exist.



(CSQ) Controller Set Quick

Use this command to move a controller position to a new absolute value quicker than using the CS command. The CS command may take up to 40ms before responding with an ACK because the unit is verifying that the specified <CONTROLLER NUMBER> exists in the site. The CSQ response will be sent with 5-10ms for a single operation and within 2ms each for single burst operations. The command must specify the controller number and the new controller position. The trade off is that the CSQ command will not check to see if the specified <CONTROLLER NUMBER> exists in the site. **Note:** It will always respond with an ACK even if it does not. If your control system does not need to check that the controller number was correctly set, the CSQ command may be used without any downside.

The syntax of the command is:

CSQ <CONTROLLER NUMBER> <CONTROLLER POSITION><CR>

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer, and <CONTROLLER POSITION> is a 16-bit number in decimal (0-65535).

The device will respond with the string whether it recognizes the controller number or not:

ACK<CR>

(CC) Change Controller

Use this command to move a controller to a new relative value. This command will increment or decrement a controller by a specified amount. The command must specify the controller number, whether it should be incremented or decremented, and the amount to change by. The syntax of the command is:

**CC <CONTROLLER NUMBER>
<DEC/INC> <AMOUNT><CR>**

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer, <DEC/INC> is 0 to decrement and 1 to increment, and <AMOUNT> is the amount to increment or decrement (a decimal number, 0-65535). If the amount to be decremented or incremented causes the parameter to exceed its minimum or maximum value, the value will be limited to its minimum or maximum value. For example, if you increment a parameter by 10 and its current value is 65530, the new value will be limited to 65535.

If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the specified controller number does not exist.

(GS) Get Controller

This command will return the controller position (value) associated with a specific controller number. The command must specify the controller number. The syntax of the command is:

GS <CONTROLLER NUMBER><CR>

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer.

If the command is accepted, the unit will respond with the string:

<CONTROLLER POSITION><CR>

Where controller position is a 16-bit number in decimal (0-65535)

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the specified controller number does not exist.

If the value being requested is a button that only has two states, the returned values will be either 0 or 65535, regardless of the actual value sent to the controller. For example, assume controller number 1 controls a mute button. If you send

CS 1 754 <CR>

and then

GS 1 <CR>

the response will be

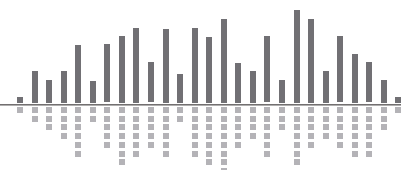
0<CR>

not

754<CR>.

More generally, if the parameter you are controlling has granularity coarser than the 16-bit values used, the returned values will be quantized to the granularity of the parameter. Controls where you might observe this effect are buttons as mentioned above and input selectors.

If the same controller number is controlling multiple parameters in Composer, only one acknowledgement value is returned from the system.



(GS2) Get Controller with Controller Number

This command will return the controller number with controller position (value) associated with it together in the same string. This command is provided at the request of third-party control programmers, including AMX/Creston programmers, to make it easier to interpret and parse returned controller positions. The command must specify the controller number. The syntax of the command is:

GS2 <CONTROLLER NUMBER><CR>

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer.

If the command is accepted, the unit will respond with the string:

<CONTROLLER NUMBER> <CONTROLLER POSITION><CR>

Where controller position is a 16-bit number in decimal (65535).

For example, sending the command

GS2 368 <CR>

Will return the response in the format below when the parameter value is 47114:

368 47114 <CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the specified controller number does not exist.

(GPR) Get Preset

This command will return the last preset that was loaded. The syntax of the command is:

GPR<CR>

If the command is accepted, the unit will respond with the string:

<PRESET NUMBER><CR>

The <PRESET NUMBER> return value will be 0-1000. A return value of 0 indicates that no preset has been recalled. The value for the preset number will always be 4 digits, with leading zeros added as necessary (e.g. 7 would be returned as 0007).

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

(GSB) Get Controller Block

This command will return the controller position (value) of a specific range of consecutive controller numbers. The command must specify the starting controller number and the number of consecutive controllers to return. The syntax of the command is:

GSB <CONTROLLER NUMBER> <BLOCK SIZE><CR>

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer and <BLOCK SIZE> is the number of consecutive controllers. Note that <BLOCK SIZE> can be at most 256.

If the command is accepted, the unit will respond with the string:

<CONTROLLER POSITION1><CR>

<CONTROLLER POSITION2><CR>

<CONTROLLER POSITION3><CR>

...

<CONTROLLER POSITIONn><CR>

Where <CONTROLLER POSITIONn> is a 16-bit number in decimal (0-65535), or -1 if a controller does not exist. The values will always be five digits, with leading zeros added as necessary (e.g. 7 would be returned as 00007<CR> and -1 would be returned as -0001<CR>.)

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the requested block size is larger than 256. For more information and tips on reading back controller numbers, see the GS command.

Example command sent:

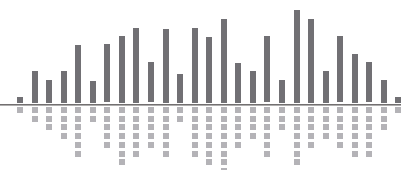
GSB 9 3<CR>

Example Response:

32321<CR>

00256<CR>

00003<CR>



(GSB2) Get Controller Block with Controller Number

This command will return the controller number with controller position (value) associated with it for a specific range of consecutive controller numbers. The command is very similar to GSB described above, but the return string may be easier to process in some systems. The command must specify the starting controller number and the number of consecutive controllers to return. The syntax of the command is:

GSB2 <CONTROLLER NUMBER> <BLOCK SIZE><CR>

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer and <BLOCK SIZE> is the number of consecutive controllers. Note that <BLOCK SIZE> can be at most 256.

If the command is accepted, the unit will respond with the string:

```
#<CONTROLLER NUMBER1>=
<CONTROLLER POSITION1><CR>

#<CONTROLLER NUMBER2>=
<CONTROLLER POSITION2><CR>

#<CONTROLLER NUMBER3>=
<CONTROLLER POSITION3><CR>

...

#<CONTROLLER NUMBERn>=
<CONTROLLER POSITIONn><CR>
```

Where <CONTROLLER NUMBERn> is the decimal controller number (1-10000) assigned in Composer and <CONTROLLER POSITIONn> is a 16-bit number in decimal (0-65535), or -1 if a controller does not exist. The values for the controller number and position will always be five digits, with leading zeros added as necessary (e.g. 7 would be returned as 00007 and -1 would be returned as -0001).

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the requested block size is larger than 256. For more information and tips on reading back controller numbers, see the GS command.

Example command sent:

GSB2 9 3<CR>

Example Response:

```
#00009=32321<CR>
#00010=00256<CR>
#00011=00003<CR>
```

(GSB3) Get Controller Block with Controller Number

This command will echo the command and then return the controller number with controller position (value) associated with it for a specific range of consecutive controller numbers. The command is very similar to GSB2 described above, but the return string includes the original command followed by the GSB2-style response. The syntax of the command is:

GSB3 <CONTROLLER NUMBER> <BLOCK SIZE><CR>

Where <CONTROLLER NUMBER> is the decimal controller number (1-10000) assigned in Composer and <BLOCK SIZE> is the number of consecutive controllers. Note that <BLOCK SIZE> can be at most 256.

If the command is accepted, the unit will respond with the string:

```
GSB3 <CONTROLLER NUMBER> <BLOCK SIZE> <CR>

#<CONTROLLER NUMBER1>=
<CONTROLLER POSITION1><CR>

#<CONTROLLER NUMBER2>=
<CONTROLLER POSITION2><CR>

...

#<CONTROLLER NUMBERn>=
<CONTROLLER POSITIONn><CR>
```

Where <CONTROLLER NUMBERn> is the decimal controller number (1-10000) assigned in Composer and <CONTROLLER POSITIONn> is a 16-bit number in decimal (0-65535), or -1 if a controller does not exist.

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the requested block size is larger than 256.

Example command sent:

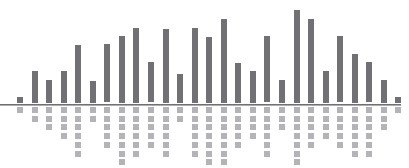
GSB3 140 3<CR>

GSB3 00140 00003<CR>

00140=00000<CR>

00141=65535<CR>

00142=32768<CR>



(LP) Load Preset

This command will load the specified preset (1-1000). The syntax of the command is:

LP <PRESET NUMBER><CR>

Where <PRESET NUMBER> = 1-1000

If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the specified preset has not been defined.

(FU) Flash Unit LEDs

This command momentarily flashes the front panel LEDs of the unit. This command can be used as a quick test to verify communications. The syntax of the command is:

FU [<cycles>] <CR>

<cycles> indicates the number of times to cycle the LEDs on and off. If <cycles> is set to zero, it halts the flashing immediately. If not specified, it defaults to 8 cycles.

If the command is accepted, the LEDs will flash and the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

(SSYSS) Set System String

This command sets a system string such as a speed dial name or number. The syntax of the command is:

**SSYSS <unit>.<resource>.<enum>.<card>.<channel>
=[<value>]<CR>**

Sets the string resource defined by the 5 parameters above to <value>. Supported values for <resource> are shown in the following table:

Resource Number	Resource Name
1000	Speed dial number
1001	Speed dial name
1004	Number to dial (VoIP only)

<unit> is the unit enumerator after the dash shown in Composer above each unit, e.g. "Edge-1" means <unit> = 1.

<enum> is zero based, 0-19 for speed dials 1-20.

<card> is 0-3 for plug-in slots A-D.

<channel> is zero based and 0 where not applicable.

The following example sets the name of speed dial #3 on card A of unit 1 to 'Acme Inc.'

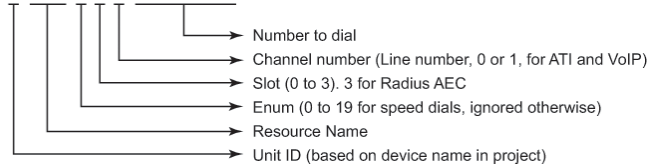
SSYSS 1.1001.2.0.0=Acme Inc.

This example sets the number of speed dial #20 on card D of unit 1 to '555-1234'.

SSYSS 1.1000.19.3.0=555-1234

Below is an example of using the SSYSS command to dial telephony digits directly.

SSYSS 1.1004.0.3.0=14257787728



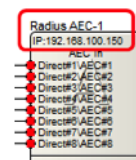
If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

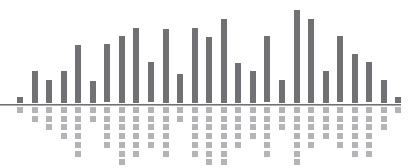
As an example, consider a Radius AEC that is labeled as Radius AEC-1 as shown in the following figure. The "-1" at the end of the unit name indicates this is unit ID 1.



To set the speed dial name and number of location 12, to "Conference Rm 1" and "555-1234", respectively, use the following commands:

SSYSS 1.1001.11.3.0=Conference Rm 1<CR>

SSYSS 1.1000.11.3.0=555-1234<CR>



And to dial Speed Dial 12, use the following command assuming the speed dial 12 button has control number 112 assigned to it:

CS 112 65535<CR>

To enter a number to dial directly by entering the digits 14257787728, send the command

SSYSS 1.1004.0.3.0=14257787728<CR>

And to dial the digits, send the following command assuming the connect/disconnect has control number 140 assigned to it:

CS 140 65535<CR>

Dialing VoIP Strings

When using the 1004 resource to dial strings with the VoIP interface, the maximum length phone number that may be directly dialed is 63 characters. Valid characters include 0-9, #, *, ., and letters (for SIP URI dialing). The digit pause character ',' to generate 2 seconds of pause in a dialing string is not supported for the VoIP interface. Direct dial is not supported for the ATI card.

(GSYSS) Get System String

This command gets a system string such as a speed dial name or number. The syntax of the command is:

GSYSS <unit>.<resource>.<enum>.<card>.<channel> <CR>

Gets the string resource defined by the 5 parameters above. Supported values for <resource> are shown in the following table:

Resource Number	Resource Name
1000	Speed dial number
1001	Speed dial name
1002	Dialed number
1003	Caller ID
1004	Number to dial (VoIP only)
1005	Call status (VoIP only)
1006	Elapsed time in a call (VoIP only)

<unit> is the unit enumerator after the dash shown in Composer above each unit, e.g. "Edge-1" means <unit> = 1.

<enum> is zero based, 0-19 for speed dials.

<card> is 0-3 for A-D.

<channel> is zero based and 0 where not applicable.

For example:

GSYSS 1.1001.11.3.0<CR>

Returns the name of speed dial #12 on unit 1, card D. No channel specification is necessary.

GSYSS 2.1003.0.3.1<CR>

Returns the caller ID on unit 2 for line 2 on card D.

If the command is accepted, the unit will respond with string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

Other examples from an existing system are shown below:

GSYSS 1.1003.0.3.0<CR>

Returns: 710 'Room 710' 4/15 7:15

GSYSS 1.1005.0.4.0<CR>

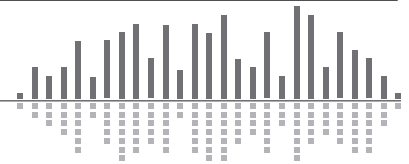
Returns: In Call: 710

GSYSS 1.1006.0.3.0<CR>

Returns: Time 0:00:35

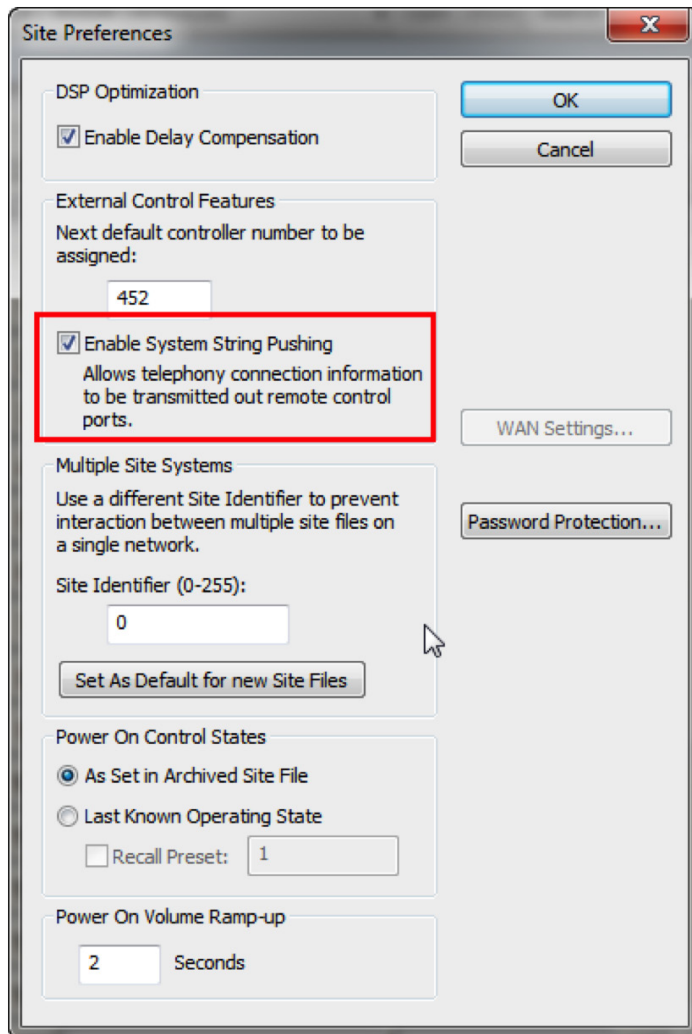
Valid values of Resource Number 1005 are shown in the following table. The parameter <dialed number> represents the digits that were dialed, if any, to initiate the call.

Telephony Interface	Status Message	Description
VoIP Interface	Busy:	The call appearance is busy
	Dialing:	The call appearance is being dialed.
	Hold:	The call appearance is in hold mode
	Idle	The call appearance is in idle state.
	In Call:	The remote caller has answered and a call is connected.
	Remote Hold:	The remote caller has placed the call on hold.
	Ringing	The call appearance is ringing.
	Unavailable:	The call appearance cannot connect.
Analog Telephone Interface	Reset	The telephone line has been reset.
	Ready	The line onhook and ready for a call.
	Outgoing Call	The line is dialing a call
	Ringing	The line is actively ringing.
	Ring Silence	The line is quiet between active rings.
	Incoming Call	An incoming call has dialed the telephony interface.
	Disconnect	The call has been disconnected.
	In Use	The telephone line is already in a call – perhaps due to a shared line with another phone.
	No Line	There is no analog phone line connected.



Pushing GSYSS String Values

To have changes in the GSYSS string values automatically sent to the control system, enable the option “Enable System String Pushing” under the Tools->Site Preferences menu and re-push the configuration file to the device. Afterwards, any changes in the values will be automatically sent to the control ports.



Push Commands

Units can send out unsolicited RS-232 and Ethernet data as a result of parameter values changing within the unit.

All parameters that can be externally controlled can be configured to automatically send out their values whenever they change. This method, referred to as pushing data, can be used instead of, or in addition to, polling (asking for data).

When using this feature, ensure that your system can handle the volume of data you set up and that it can differentiate between responses to commands and pushed data.

Commands used to control the push feature are described below. Also, the following questions and answers provide a detailed discussion of this feature, including real-world problems and solutions.

When is data pushed?

For data to be pushed

- 1) a controller number must be enabled for push in Composer
- 2) individual parameters must be enabled to push using the Push Enable command.

Then, the controller value will be sent out

- 1) whenever the control's underlying parameter changes or
- 2) when a refresh command is issued via RS-232 or Ethernet.

Regardless of if the parameter change is made via the control application, RS-232, RS-485, preset recall, analog control or any other method, the data will be pushed. This means for example that if your control system changes a controller value set up for push, you will immediately receive notification of that change.

Where (out what port) is the data pushed?

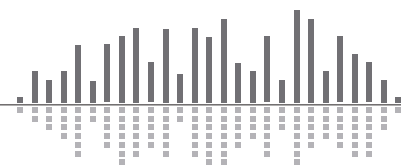
The data is sent out the RS-232 port and the Ethernet port (both UDP and TCP) of the Symetrix unit.

What does the pushed data look like?

The format for unsolicited or “push” data is the same as the GSB2 command. Strings consist of the controller number and its value in the following format:

#<CONTROLLER NUMBER>=<CONTROLLER POSITION><CR>

Where **<CONTROLLER NUMBER>** is the decimal controller number (1-10000) assigned in Composer and **<CONTROLLER POSITION>** is a 16-bit number in decimal (0-65535). The values for the controller number and position will always be five digits, with leading zeros added as necessary (e.g. 7 would be returned as 00007).



Up to 64 strings, separated with a <CR> as shown, may be sent out together.

Example:

#00007=12321<CR>

#00324=00128<CR>

#10000=65535<CR>

Should I use the push feature or poll for parameter changes?

The decision is up to you. Use whichever method makes more sense for your application and control system. Manual polling is often simpler to implement initially because all data from the unit is a direct response to command you send it, simplifying parsing. However, in situations where a large number of parameters that change infrequently need to be monitored, pushing may make more sense. You may also prefer the convenience of not needing to set up a timer to continually poll parameters for changes. Use whatever method is appropriate for your situation.

How often is data pushed?

If there is data to be pushed, it is normally sent out every 100 milliseconds. This is called the push interval. While 100 ms is the default, the push interval can be changed via a Set Push Interval (PUI) command.

Can I push meter data?

Yes, meters can be enabled for push. Keep in mind that with normal audio signals connected to a meter, the meter value will most likely be changing constantly, so you will typically see the meter data being pushed at every 100 ms interval. However, a Set Push Threshold command can be used to prevent pushes until the data differs by a specified amount (by default, this amount is 1). Another option is to use an LED display, peak detector, and/or threshold detector modules instead of directly pushing the meter data.

How can I control the amount of data pushed?

There are several methods for controlling pushed data. First, since pushed data is enabled on a per-control basis, your first line of defense is to limit it to only certain controls. Second, pushing can be globally turned on and off using this protocol. Third, pushing can be enabled for just a range of controller numbers. Fourth, the Set Push Threshold command can be used to prevent pushes until the data changes by a specified amount. Fifth, the Set Push Interval command controls how often the data is pushed, useful for meters and other data that changes frequently. Finally, the Push Refresh and Push Clear commands provide additional methods of control.

I want to refresh everything to make sure my control system is synchronized to the hardware. How can I receive all data even if it hasn't changed?

Use the Push Refresh (PUR) (GC) command. Alternatively, you could use the Get Controller commands to manually ask for the controls you are concerned with.

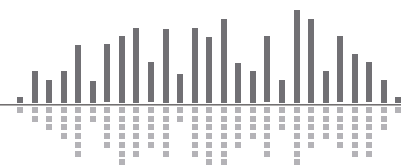
Sometimes my control system turns off push for an extended period of time. When I turn it on, will I be notified of all changes that occurred while push was turned off?

Yes, by default, all changes made while push was off will be immediately reported as soon as it is turned on. This applies to both turning push off globally or for individual controllers via the Push Disable command. Take care that your system can handle the potentially large amount of data that can be generated. It may be helpful to “gradually” turn on the push feature, enabling a small range of controller numbers at once. You can also use the Push Clear command to deal with this scenario. It allows you to effectively ignore all previous unreported changes.

What is the difference between the Global Push Enable/Disable (PU) command and the Push Enable (PUE) and Push Disable (PUD) commands? Why are there 2 different ways to specify a range of controllers?

The Global Push Enable/Disable command can be used to completely turn off push, or turn on push for all or a single contiguous range of controller numbers. In contrast, the Push Enable/Disable command allows much finer control. Individual (non-contiguous) controllers can be turned on and off, hence multiple ranges are supported.

The reason both methods are provided is for backwards compatibility. The less flexible “single range” global PU command was added first. Later, the more flexible PUE and PUD commands were added as an enhancement. The older global method was left in so existing programs wouldn't need to be modified. We recommend that you use either one system or the other exclusively. Do not combine them. New designs should use the PUE and PUD commands and never use the PU command with a range specified.



What is the difference between the Push Enable (PUE) and Push Disable (PUD) commands and the “Enable Push” button in Composer?

Both of these methods can be used to enable or disable an individual controller for push. The difference is that changing the setting in Composer requires you to push to the design to hardware again, whereas the Push Enable and Push Disable commands take effect immediately and are intended to be used for “on the fly” changes. In addition, Composer settings are permanent across power cycling, whereas changes made with the Push Enable and Push Disable commands are reset by a power cycle (all controllers default to enabled). Please note that the “Enable Push” button acts as an override so it must be enabled for a controller ever to push. You cannot issue a Push Enable command for a parameter that doesn’t have the “Enable Push” setting active.

How does push work at power-up?

When a unit is first powered up, push is globally turned on and all controllers are individually enabled. All controller numbers are assumed to have changed. This means that after power-up, you will immediately receive all current values.

What about pushed system strings?

The pushing of system strings enabled/disabled in the Site Preferences. When enabled, any changes made to speed dials, etc. will be pushed out the RS-232 port and the Ethernet port of the Symetrix unit. They also push on a Site File push and unit power up.

I’m not receiving unsolicited data. Any suggestions for troubleshooting?

First of all, make sure that general communication is working between your control system and the Symetrix RS-232 port or Ethernet port. Make sure you can send commands and receive ACK messages. Try the Flash Unit (FU) command.

Make sure the parameters you want pushed have the “Enable Push” setting active in Composer. You can verify this using the Controller Manager, and making sure there is a ‘Yes’ in the Push column. Be sure you Go On-Line (Push Design To Hardware) or F4 after making any changes in Composer.

For Ethernet, make sure the Ethernet port is connected to the same network as the control system. Verify the connection LED on the Ethernet jack and/or switch is lit. Verify you can “ping” the unit using its IP address.

Make sure the push feature has been globally enabled using the Global Push Enable/Disable command. Push is globally enabled on power-up, but may be turned off via RS-232 or Ethernet. Power cycling the unit is a quick way to verify this.

Make sure the individual controllers have been enabled using the Push Enable command. Push is disabled for all controllers on power-up, and must be turned on via RS-232 or Ethernet. Sending a PUE command is a quick way to enable all controllers.

Make sure the parameter to be pushed is changing. Change the parameter via the control application, a Controller Set command, or other method. You can also use the Push Refresh command to force the data to be sent. If you have changed the push threshold, make sure the parameter is changing by an amount larger than the threshold.

For Ethernet, the unit needs to know the proper IP address to send the data. Make sure at least one command has been sent from the control system to the unit. If the control system ever changes IP addresses, another command must be sent to establish the new address.

What are the limitations of this feature?

If multiple parameters change at the same time, up to 64 controller numbers will be sent out during each push interval (default 100 ms) until all have been sent out. This maximum rate may be further limited by your RS-232 baud rate. If a large amount of data is being pushed, we recommend using the highest baud rate your system can support. A baud rate of 115200 is sufficient to prevent further limiting of the throughput.

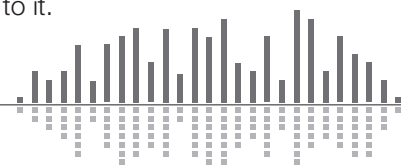
Commands Related to Push:

(PU) Global Push Enable/Disable

This command enables or disables the push feature. When enabling, a range of controllers can be specified to allow pushing only certain values. Disabling is always global and prevents any unsolicited data from being pushed. The syntax of the command is:

PU <ON/OFF> [<LOW> [<HIGH>]]<CR>

Where <ON/OFF> is 0=OFF and 1=ON, <LOW> is the optional lowest controller number to push (only valid when enabling) and <HIGH> is the optional highest controller number to push (only valid when enabling). <LOW> and <HIGH> are both decimal controller numbers (1-10000) assigned in Composer. If no controller numbers are specified, the entire range of 1-10000 will be enabled for push. If only one controller number is specified, it is assumed to be the <LOW> value and the range from that number, up to 10000, will be pushed. If two controller numbers are specified, the range formed by those values (including the values themselves) will be enabled for push. <LOW> must be less than or equal to <HIGH>. When enabling, the range specified overrides any previous ranges, i.e. it replaces the range, rather than adding to it.



If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

At power-on, push is always enabled. Remember that individual controller numbers must be enabled using the **Push Enable** command as well. Data is pushed whenever a change in that controller occurs or if forced to refresh using the **Push Refresh** command.

Note: **Global Push Enable** with a range specified, e.g. **PU 1 100 200<CR>** is not recommended. Instead, we recommend always globally enabling the entire range using **PU 1<CR>** and using the **Push Enable** command for individual control.

(PUE) Push Enable

This command enables the push feature for an individual controller or range of controllers. The syntax of the command is:

PUE [<LOW> [<HIGH>]]<CR>

Where **<LOW>** is the optional lowest controller number to push and **<HIGH>** is the optional highest controller number to push. **<LOW>** and **<HIGH>** are both decimal controller numbers (1-10000) assigned in Composer. If no controller numbers are specified, the entire range of 1-10000 will be enabled for push. If only one controller number is specified, only that controller number is enabled. If two controller numbers are specified, the range formed by those values (including the values themselves) will be enabled for push. **<LOW>** must be less than or equal to **<HIGH>**. Multiple PUE commands can be used to enable non-contiguous controller numbers since changes are additive.

If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

At power-on, push is enabled for all controllers in Composer systems. Remember that individual controller numbers must have the **Enable Push** setting active in Composer. Data is pushed whenever a change in an enabled controller occurs or if forced to refresh using the **Push Refresh** command. Changes that happen while a control is disabled will be pushed immediately upon enabling that control. The **Push Disable** command is the inverse of this command and provides a way to turn off controllers for push.

(PUD) Push Disable

This command enables the push feature for an individual controller or range of controllers. The syntax of the command is:

PUD [<LOW> [<HIGH>]]<CR>

Where **<LOW>** is the optional lowest controller number to stop pushing and **<HIGH>** is the optional highest controller number to stop pushing. **<LOW>** and **<HIGH>** are both decimal controller numbers (1-10000) assigned in Composer. If no controller numbers are specified, the entire range of 1-10000 will be disabled for push. If only one controller number is specified, only that controller number is disabled. If two controller numbers are specified, the range formed by those values (including the values themselves) will be disabled for push. **<LOW>** must be less than or equal to **<HIGH>**. Multiple PUD commands can be used to disable non-contiguous controller numbers since changes are subtractive.

If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

At power-on, push is enabled for all controllers in Composer systems. Remember that individual controller numbers must have the **Enable Push** setting active in Composer. The **Push Enable** command is the inverse of this command and provides a way to turn on controllers for push.

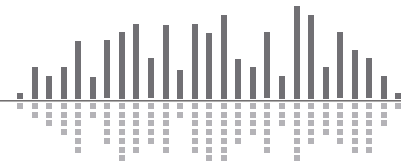
(GPU) Get Push-enabled Controllers

This command returns a list of all controllers currently enabled for push. A range may optionally be specified to limit the display to controllers enabled for push within that range. The syntax of the command is:

GPU [<LOW> [<HIGH>]]<CR>

Where **<LOW>** is the optional lowest controller number to inquire about and **<HIGH>** is the optional highest controller number to inquire about. **<LOW>** and **<HIGH>** are both decimal controller numbers (1-10000) assigned in Composer. If no controller numbers are specified, the entire range of 1-10000 will be inquired about. If only one controller number is specified, it is assumed to be the **<LOW>** value and the range from that number up to 10000 will be inquired about. If two controller numbers are specified, the range formed by those values (including the values themselves) will be inquired about.

<LOW> must be less than or equal to **<HIGH>**.



If the command is accepted, the unit will respond with a list of enabled controller numbers separated by **<CR>**. If no controllers are enabled, it returns the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

Special case: Entering **GPU 0<CR>** will return a list settings related to push. It begins with **Global=<0/1>** to show if push is globally enabled (1) or disabled (0).

This is followed by five 5-digit values showing the settings of

- 1) the global lower limit,
- 2) the global upper limit,
- 3) the threshold for parameters,
- 4) the threshold for meters, and
- 5) the push interval in milliseconds.

The default printout would look like this:

Global=1<CR>

00001 10000 00001 00001 00100<CR>

(PUR) Push Refresh

This command causes data to be pushed immediately even if it hasn't changed (assuming push is enabled). This may be useful when trying to synchronize a control system to the unit. A range of controllers can be specified to refresh only certain values. The syntax of the command is:

PUR [<LOW> [<HIGH>]]<CR>

Where **<LOW>** is the optional lowest controller number to refresh and **<HIGH>** is the optional highest controller number to refresh. **<LOW>** and **<HIGH>** are both decimal controller numbers (1-10000) assigned in Composer. If no controller numbers are specified, the entire range of 1-10000 will be refreshed. If only one controller number is specified, it is assumed to be the **<LOW>** value and the range from that number up to 10000 will be refreshed. If two controller numbers are specified, the range formed by those values (including the values themselves) will be refreshed. **<LOW>** must be less than or equal to **<HIGH>**.

If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

At power-on, all controller values are assumed to have

changed, so it acts as if a full refresh was performed. Remember that individual controller numbers must have the Enable Push setting active in Composer. In addition, push must be enabled for the range of controllers you are refreshing (see **Push Enable/Disable**). Controller numbers that don't meet this criterion will not be affected by the **Push Refresh** command. In other words, if a controller is not enabled for push, refreshing it won't cause the value to be pushed even if that controller is later enabled. The controller must be enabled for push at the time the Push Refresh command is issued.

(PUC) Push Clear

This command causes previous changes in data to be ignored and not pushed. It may be desirable to issue this command when first enabling push to prevent being swamped by the flood of incoming data. A range of controllers can be specified to clear only certain values. The syntax of the command is:

PUC [<LOW> [<HIGH>]]<CR>

Where **<LOW>** is the optional lowest controller number to clear and **<HIGH>** is the optional highest controller number to clear. **<LOW>** and **<HIGH>** are both decimal controller numbers (1-10000) assigned in Composer. If no controller numbers are specified, the entire range of 1-10000 will be cleared. If only one controller number is specified, it is assumed to be the **<LOW>** value and the range from that number up to 10000 will be cleared. If two controller numbers are specified, the range formed by those values (including the values themselves) will be cleared. **<LOW>** must be less than or equal to **<HIGH>**.

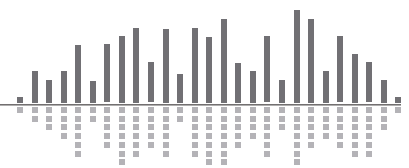
If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

It may be useful to issue this command if push has been disabled for a long time and then is about to be re-enabled. Otherwise, you will immediately receive notification for all changes that occurred during the disabled time.



(PUI) Set Push Interval

This command changes the minimum length of time between consecutive pushes of data. (See “How often is data pushed?” above for more information.) At power-up, this value defaults to 100 milliseconds. The syntax of the command is:

PUI <MILLISECONDS><CR>

where <MILLISECONDS> is the push interval in milliseconds, between 20 ms and 30,000 ms (30 seconds).

If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

While setting a short interval can speed up the push response, it may have a negative impact on overall system performance. The shorter the interval, the more time will be spent looking for push data. This can slow down responses to other RS-232 commands and Composer. Therefore, we recommend using the longest interval that is practical, especially if data is being pushed while Composer is on-line. The default value of 100 milliseconds usually provides a good compromise between prompt reports of changing data and overall system performance.

Note that in cases where many controllers are changing rapidly, the serial baud rate may ultimately limit the update rate. Using the highest possible baud rate is recommended.

(PUT) Set Push Threshold

This command changes the push threshold value. Recall that data is only pushed when it changes. The threshold is the amount a value must change from the previous push before it is pushed again. For example, if a controller value was 10,000 and the threshold was 1,000, the data would not be pushed again until the value rose to at least 11,000 or fell to 9,000 or below.

Units maintain two different thresholds: one for parameter data such as faders and buttons, and another for meters (including LEDs). These two thresholds can be set to the same value or be different. It may be desirable to use a fairly large threshold for meters to avoid constant pushing of values. The power-on default for both of these values is 1.

The syntax of the command is:

PUT [<PARAMETER THRESH>] [<METER THRESH>]<CR>

Where <PARAMETER THRESH> is the optional threshold for parameters other than meters (e.g. faders and buttons) and <METER THRESH> is the optional threshold for meters. Both values must be between 0 and 65535. If neither threshold is specified, both thresholds are set to the default of 1. If only one threshold is specified, that value is used for both the parameter and meter thresholds.

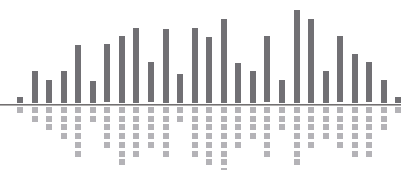
If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

Technical Note: The threshold is a “greater than or equal to” type parameter, meaning it must be met (or exceeded) to trigger a push. For example: if the threshold is 1 and the last value pushed was 10,000, then a new value of 10,001 or 9,999 would cause a push to occur. A threshold of 0 acts just like 1.



Setup Commands

The defaults for the **Set Baud** and **Set Echo Mode** commands are configured in Composer via the *Unit Properties* dialog. Settings made via these Control Protocol commands will be overridden by the values in the last pushed Site File on the next hardware reboot or push. To establish the permanent state, right-click on each unit and select *Unit Properties*, then click *Configure Remote Control Port(s)*. Settings may be set independently for the Ethernet and RS-232 ports.

Note: If you ever find yourself in a situation where you are not sure of the RS-232 port settings, you can use Composer to return the unit to default values. Simply open the Hardware->Upgrade Firmware/Hardware Settings dialog and click on the Erase Memory button. You can then select to erase the Accessory Port Settings. Doing this will set the baud rate, echo mode and quiet mode settings to the factory default values.

Alternatively, the rear panel reset button can be used to return the settings to factory defaults. However, that should be only used as a last resort since it also resets many other settings.

(SB) Set Baud

The Set Baud command controls the RS-232 port's baud rate.

The syntax of the command is:

SB <BAUD><CR>

Where **<BAUD>** is 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200.

If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

The baud rate is saved in non-volatile memory so it is retained across power cycles. This command can be sent to either the Ethernet or RS-232 port, but only affects the RS-232 port.

Note: New units default to a baud rate of 57600.

(SQ) Set Quiet Mode

The Set Quiet Mode command controls the text output of the control port during responses. When quiet mode is turned on, it restricts the output to just ACK, NAK or simple values. All command descriptions above assume that quiet mode is turned ON. Quiet mode ON should generally be used for normal operation.

When quiet mode is set to OFF, lengthy strings intended to be read by humans are sent in response to commands. This mode is useful when using a terminal program for testing or debugging.

The syntax of the command is:

SQ <ON/OFF><CR>

Where **<ON/OFF>** is 0 = OFF, 1 = ON.

If the "SQ 0" command is accepted, the unit will respond with the string:

Setting Quiet Mode to false.<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

The quiet mode state is kept separately for the Ethernet port and the RS-232 port. A Set Quiet Mode command received on the Ethernet port does not affect the RS-232 port and vice versa. The quiet mode state is saved in non-volatile memory, so it is retained across power cycles.

Note: New units default to quiet mode ON.

(EH) Set Echo Mode

The Set Echo Mode command controls the text output of the control port during commands. When echo mode is turned on, all characters that are received on the RS-232 or Ethernet port are sent or "echoed" back. This mode is useful when using a terminal program for testing or debugging.

When echo mode is turned off, the characters received are not echoed back. All command descriptions above assume that echo mode is turned off. Echo mode OFF should generally be used for normal operation.

The syntax of the command is:

EH <ON/OFF><CR>

Where **<ON/OFF>** is 0 = OFF, 1 = ON.

If the command is accepted, the unit will respond with the string:

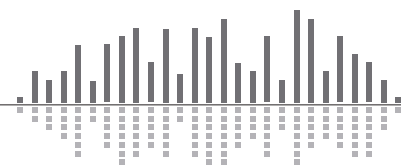
ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

The echo mode state is kept separately for the Ethernet port and the RS-232 port. A Set Echo Mode command received on the Ethernet port does not affect the RS-232 port and vice versa. The echo mode state is saved in non-volatile memory so it is retained across power cycles.

Note: New units default with echo mode OFF.



System Commands

NOP (No Operation)

The NOP command will always return an ACK in Quiet mode to indicate that the system has received the command. The NOP command can be used to keep a TCP connection active or as a simple check of whether a system is up and running and responding to commands.

Q! (Quit TCP/IP Session)

The Q! command will close the current TCP/IP session. If the command is accepted, the unit will respond with the string:

ACK<CR>

If the command is interpreted but fails for any reason the unit will respond with the string:

NAK<CR>

A typical reason for failure is that the command was not sent over a TCP/IP connection.

RI (IP address)

The RI command returns the IP address of the unit.

As an example:

RI<CR>

192.168.100.150

V (Version)

The V command returns the version of the software in the unit. Up to thirteen different versions of software will be returned corresponding to different subsystems within the devices. For this command to be useful, it is recommended to run the command while verbose mode is turned on or temporarily turned on with the \$v command prefix (described in Special Command Prefix section).

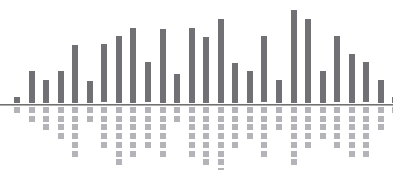
As an example:

\$v V <CR>

```
Firmware Version:  V3.000
Build Date:       Tue Apr  8 12:36:30 PDT 2014
Internal Build:   V8.00
Linux Version:    V3.2.0
Dante firmware:   F:3.6.4 Sym:5.42
Startup firmware: V2.18
Front panel firmware: V2.29
RS-485 firmware:  V1.25
RS-232 firmware:  V1.55
TCP/IP firmware:  V1.02
ARC-Web firmware: V1.10
ARC-Web executable: V1.09
ARC-Web CGI firmware: V1.09
Dialer CGI firmware: V1.55
Additional Info:
Linux version 3.2.0-ADI-2012R1-pre-9-pae+
(uclinux-devel@linux-tclw)
(gcc version 4.3.5 (ADI-2010R1-RC4) ) #3119 Tue
Nov 19 17:13:55 PST 2013 TCP/IP instance #1 of 4
>
```

R! (Reboot device)

The R! command will reboot the unit.



Special Command Prefixes

Special command prefixes, added in Composer 3.0, allow the programmer to temporarily enable the echo mode, verbose mode, or quiet mode for that particular command. The \$v prefix is helpful when running in Quiet mode and executing the V command.

\$e (Echo Mode)

The \$e prefix will execute the command and echo back the command that was sent. This can be helpful when correlating a response to the commands that generated it.

The echoed command will be enclosed in {curly brackets}.

As an example:

\$e CS 140 65535<CR>

Causes the following acknowledgement

{CS 140 65535} ACK

\$v (Verbose Mode)

The \$v prefix will execute the command as if quiet mode was turned off for the duration of the command.

As an example:

\$v SSYSS 1.1004.0.3.0=14257787728 <CR>

will return the string

>Set System String '1.1004.0.3.0' to '14257787728' succeeded>

As another example:

GSYSS 1.1001.11.3.0 <CR>

could return the string

Conference Room

\$v GSYSS 1.1001.11.3.0 <CR>

could return the string

>System String '1.1001.11.3.0' is 'Conference Room'>

\$q (Quiet Mode)

The \$q prefix will execute the command as if quiet mode was turned on for the duration of the command regardless of the state of quiet mode.

As an example:

\$q SSYSS 1.1004.0.3.0=14257787728

Will return the string

ACK

Super Matrix Mixer Control Protocol

Background

Each control is represented by a number in a flat list from 1-10,000. The value of a control is a 16-bit value between 0-65535. The interpretation of that 16-bit controller value is specific to each control and each control knows how to interpret it. For example, a value of 40,000 might represent a gain of -20.7 dB in a gain module, a frequency of 1034 Hz in an EQ, and a release time of 517 ms in a compressor.

In contrast, the new system will refer to parameters in "human readable" terms, e.g. the feature name and input/output number of a matrix gain. Similarly, the value will be expressed in natural units matching the way they are shown in Composer, e.g. dB for gains. Floating point will be used. For backward compatibility, the existing 16-bit value system will also be supported.

Features to Control

The following matrix features can be controlled using the remote control protocol:

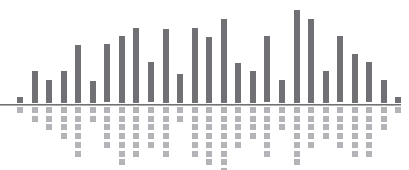
1. Crosspoint Gain
2. Crosspoint Connect status
3. Crosspoint Delay
4. Input Mute
5. Input Gain
6. Input Solo
7. Input Pan
8. Output Mute
9. Output Gain
10. Output Pre/Post

Commands

All commands take the following general format:

CMV <Action>[<Format>] <Unit>.<Module>.<Feature>.<Enumerator> [<Value>]

The first part of the command is always "CMV", which stands for "Control Module Value". The other parameters will now be explained in detail.



Action

The second argument is the action to take. It can be Set, Get, Modify, Toggle, or Reset. This chooses whether to read the current value or change the value in a variety of ways. As you might guess, “Get” reads and “Set” writes. The “Modify” option allows changing the value relative to the current setting using a positive or negative offset. The “Toggle” value toggles a boolean (button) parameter from on to off or vice versa. The “Reset” option resets a parameter to its default, i.e. what it would be in a fresh module placed in Composer.

Note that not all actions are appropriate for all feature types. For example, “Toggle” is not applicable to continuous parameters such as gains. In these cases, the command does nothing but does not return an error.

“Modify” may be used with boolean parameters, e.g. mutes and grid connect buttons. In this case, any positive modifier will turn the parameter on, any negative modifier will turn the parameter off, and a 0 modifier will do nothing.

Format

The optional <Format> field allows the user to specify the format of the <Value>. If no option is specified, it uses the native format for that particular control, i.e. dB for gains, 0/1 for Booleans, milliseconds for delay, and percentage for pans. Other options are ‘P’ or ‘%’ for percentage 0-100% and ‘L’ for the legacy 0-65535 range. The format also applies to the returned data.

Unit

The <Unit> field allows the user to specify the unit in the current site. If set to 0, the unit currently being addressed is assumed. Otherwise, it is the enumerator shown in Composer.

Module

The <Module> field allows the user to specify the module number shown in Composer. For the Super Matrix, the module number is always 1.

Feature

1. The <Feature> may be any of the following for the matrix mixer:
2. CPGain Crosspoint Gain
3. CPConnect Crosspoint Connect status
4. CPDelay Crosspoint Delay
5. IMute Input Mute
6. IGain Input Gain
7. ISolo Input Solo
8. IPan Input Pan

9. OMute Output Mute
10. OGain Output Gain
11. OPre Output Pre/Post

Enumerator

The <Enumerator> specifies which crosspoint, input, output, etc. to control. Matrix crosspoints, will be identified with an “IxOy” syntax, e.g. “I3O4” refers to input #3 output #4. For parameters that refer only to an input or output and not a crosspoint, specify just the “I” or “O” part, e.g. “I3” or “O1”. All values are 1-based so they match what the user sees in Composer.

The <Enumerator> may also be specified as a contiguous range or arbitrary set of values. If this format is used, the enumerator should be enclosed in {curly brackets}. For a range, a colon is used to separate the beginning and ends of the range. For example “{I1O1:I3O4}” specifies a 3x4 rectangle of values with upper left of Input #1 Output #1 and lower right at Input #3 Output #4.

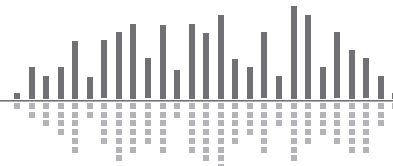
Sets of values are specified using comma-delimited lists. For example “{I1O1,I3O3,I16O12}” specifies 3 different crosspoints at input 1/output 1, input 3/output 3, and input 16/output 12.

Sets, ranges, and individual values may be combined, so complex groups may be set in a single command, e.g. “{I1O1:I3O4, I8O8:I9O9, I10O10}”. It is legal to include the {curly brackets} even if only a single value is specified. So the brackets may always be included if desired. Spaces between individual comma-separated values in a set may be used for clarity.

Value

The <Value> may be in 1 of 3 different formats as mentioned in the discussion of the <Format> field. For percent and native mode, values may be floating-point of any precision. Legacy 16-bit mode uses integers between 0-65535. In native mode, all gains are expressed in dB. Boolean parameters should be either 0 or 1. Delay values are in fractional milliseconds. In percentage mode, a percent sign may or may not be included. In dB mode, anything -72.0 dB or under will be interpreted as fully off or muted. The word “Off” may also be used. Muted values will always read back as -72.0 dB. Values larger than the accepted range will be clamped to their minimum or maximum values.

If multiple enumerators are given, all specified parameters will be set to the same <Value>. Only one <Value> may be provided in each command.



An optional equals sign may be placed in front of the value, e.g. "CMV Set 0.1.CPGain.I3O6=4.1". Doing so makes the commands and responses identical.

Basic Examples

CMV Set 0.1.CPGain.I3O6 4.1 – Set the crosspoint gain for input #3 going to output #6 to 4.1 dB.

CMV Set 0.1.CPConnect.I13O76 1 – Turns on the crosspoint for input #13 going to output #76.

CMV Toggle 0.1.OMute.O2 1 – Toggles the output mute for output #2.

CMV SetP 0.1.IGain.I12 100 – Sets the input gain to maximum (100%) for input #12.

CMV Modify 0.1.CPDelay.I1O1 -3.7 – Decrements the crosspoint delay for input #1 going to output #1 by 3.7 milliseconds.

CMV Get 0.1.CPConnect.I13O76 – Returns 0 or 1 based on the crosspoint status for input #13 going to output #76.

CMV GetL 0.1.CPVol.I3O4 – Returns a value between 0-65535 for the crosspoint gain for input #3 going to output #4.

CMV Get% 0.1.OPre.O4 – Returns 0 or 100% based on the output pre/post state of output #4.

Range Examples

CMV Set 0.1.CPGain.{I3O1:I3O20} 4.1 – Set the crosspoint gain for input #3 going to outputs #6 through 20 to 4.1 dB.

CMV Set 0.1.CPConnect.{I1O1:I3O20} 1 – Turn on the crosspoint connect for inputs #1 through 3 going to outputs #1 through 20.

CMV Toggle 0.1.CPConnect.{I5O5:I5O13} – Toggle the crosspoint connect for input #5 going to outputs #5 through 13.

CMV Modify 0.1.InGain.{I3:I17} 3 – Increase the input gain for inputs 3 through 17 by 3 dB.

CMV Get 0.1.OutGain.{O1:O5} – Returns the output gain in dB for outputs 1 through 5.

CMV Reset 0.1.OutGain.{O5:O31} – Set the output gain to the default of 0.0 dB for outputs 5 through 31.

Return Value

Get Operations

For commands that get a value, the value will be returned in the format the user specifies in <Format>. No units will be included, e.g. a gain will be returned as "-4.50" not "-4.50 dB". Assuming native mode, for gains, 2 decimal places of precision will be given. For milliseconds, 3 decimal places will be given. For Booleans, a 0 or 1 will be returned.

For pans, a percentage value with 2 decimal places will be returned.

In percentage mode, a value between 0 and 100 with 3 decimal places will be returned, e.g. "13.347%." A percent sign will be included.

In legacy 16-bit mode, a value between 0 and 65535 will be returned with no decimal places and no units.

For percentage and 16-bit mode, the scaling of all parameters is linear within. For gains, it is linear in dB, i.e. 50% would be -30 dB for a gain that ranges from -72 dB to +12 dB. For delays, 100% or 65535 is the maximum delay time supported by the current configuration.

In general, percentage and legacy 16-bit mode are discouraged because they require the control system to know the range and scaling of the parameters.

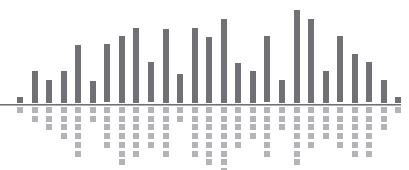
The exact format of the return is dependent on if quiet mode is engaged or not. In quiet mode, the value will simply be returned. In verbose mode, the control being set is included in the same format as a parameter is specified followed by an equals sign, then the actual value:

<Unit>.<Module>.<Feature>.<Enumerator>=<Value>

The parameter is formatted nicely so it is in a consistent case without any leading digits.

Change Operations

For commands that change a value (set, modify, toggle, reset), in quiet mode, an "ACK" or "NAK <Reason>" will be returned depending on if the command was successfully interpreted and executed. Errors include things such as unsupported options, unrecognized features, channel number out of range, or illegal values. When "NAK" is returned, the reason for the failure will be given. Reasons may be any one of the following: Action, Format, Unit, Module, Feature, Enumerator, Value, System, Other. The first 7 are self-explanatory, referring to the format of the CMV command. "System" refers to a system error, e.g. a unit is not responding. "Other" refers to an unknown issue.



In verbose, commands that change a value will respond with the changed parameter followed an equal sign and then the new value in the specified format. If multiple values are changed by a control, i.e. a range of gains, the new value for each will be returned on a separate line. Note this may generate very large volumes of traffic! It is advisable to use the quiet mode option when changing more than a handful of values to avoid this.

For values, in native mode, specifying a value outside of the defined range of the parameter is not considered an error. The value will be clipped to the minimum or maximum value supported. This is done because control systems may not understand the exact ranges of parameters. In percentage or 16-bit mode, values outside of 0-100 or 0-65535 are considered errors.

Reasons may be any one of the following: Action, Format, Unit, Module, Feature, Enumerator, Value, Configuration, System, Other. The first 7 are self-explanatory, referring to the format of the CMV command. "Configuration" means the specified feature is not supported in the current site file configuration, i.e. that feature was disabled in Composer.

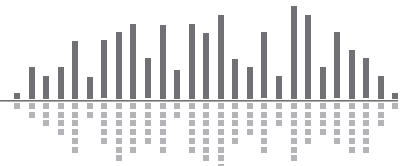
Polling for Changes

A command is available to determine if anything has changed in the matrix mixer. This allows requesting a single value rather than many individual parameters. This command is:

GSYSC <unit>.3060.-1,0

If <unit> is 0, the unit currently being addressed is assumed. Otherwise, it is the enumerator shown in Composer.

The command will return an integer value that increments each time something in the Super Matrix is changed by any mechanism (remote control, Composer, preset recall, etc.). Control systems can check if the value is different from the value last obtained to see if changes have been made. If the value has changed, it can then read back all matrix parameters of interest.



Composer Control Protocol

© 2016 Symetrix, Inc. All rights reserved. Printed in the United States of America. Mention of third-party products is for informational purposes only and constitutes neither an endorsement nor a recommendation. Symetrix assumes no responsibility with regard to the performance or use of these products. Under copyright laws, no part of this price list may be reproduced or transmitted in any form or by any means, electronic or mechanical, without permission in writing from Symetrix, Inc. If, however, your only means of access is electronic, permission to print one copy is hereby granted. The following are either Trademarks or Registered Trademarks of Symetrix, Inc.: Symetrix, SymNet Designer, SymLink and CobraLink. Windows is a Registered Trademark of Microsoft, Inc. Other product names mentioned herein may be trademarks and/or registered trademarks of other companies and are property of their respective owners.

