ANI4IN

Command Strings

Shure ANI4IN command strings for third-party control systems, such as Crestron or Extron. Includes all programming commands for block and XLR versions.
Version: 2 (2019-G)

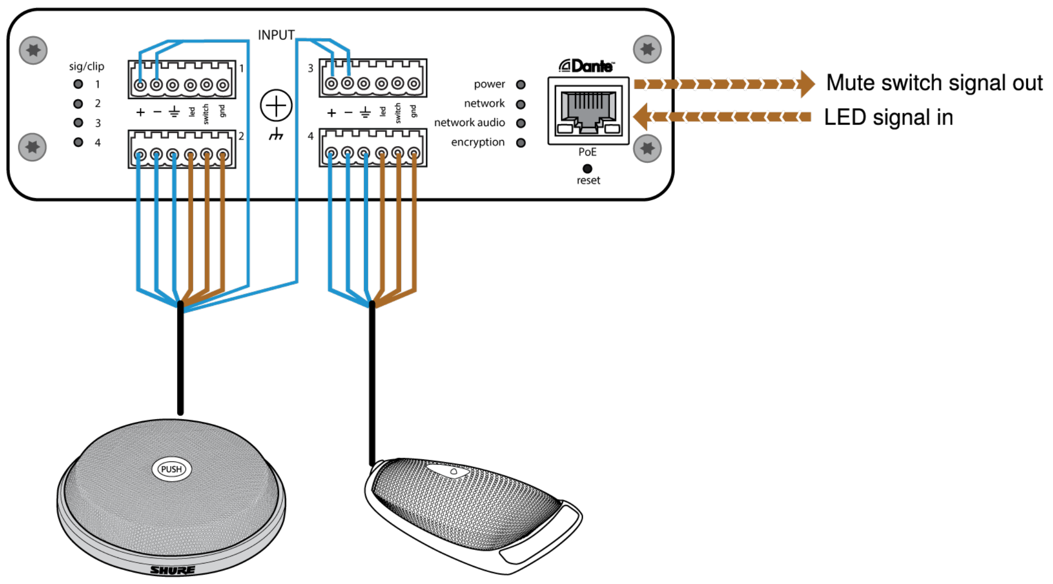
Table of Contents

ANI4IN Command Strings

Logic Applications

The block connecter model variation (ANI4IN-BLOCK) features three logic signal connections. Logic signals are converted into Ethernet command strings and sent and received by any device (such as an echo canceller or control system) that supports Ethernet command strings.

In this diagram, Shure MX392 and MX396 Microflex® microphones are connected the audio network interface. The mute button on each microphone sends a logic signal (switch) to mute other audio equipment. The microphones receive logic signals (LED) so that the microphone LED behavior reflects the state of the entire audio system.



ANI4IN Command Strings

The device is connected via Ethernet to a control system, such as AMX, Crestron or Extron.

Connection: Ethernet (TCP/IP; select “Client” in the AMX/Crestron program)
Port: 2202

Conventions

The device has 4 types of strings:

GET

Finds the status of a parameter. After the AMX/Crestron sends a GET command, the ANI4IN responds with a REPORT string

SET

Changes the status of a parameter. After the AMX/Crestron sends a SET command, the ANI4IN will respond with a REPORT string to indicate the new value of the parameter.

REP

When the ANI4IN receives a GET or SET command, it will reply with a REPORT command to indicate the status of the parameter. REPORT is also sent by the ANI4IN when a parameter is changed on the ANI4IN or through the GUI.

SAMPLE

Used for metering audio levels.

All messages sent and received are ASCII. Note that the level indicators and gain indicators are also in ASCII

Most parameters will send a REPORT command when they change. Thus, it is not necessary to constantly query parameters. The ANI4IN will send a REPORT command when any of these parameters change.

The character

“x”

in all of the following strings represents the channel of the ANI4IN and can be ASCII numbers 0 through 4 as in the following table

0	All channels
1 through 4	Individual channels

Command Strings (Common)

Get All	
Command String: `< GET x ALL >`	*Where x is ASCII channel number: 0 through 4. Use this command on first power on to update the status of all parameters.*
ANI4IN Response: `< REP ... >`	*The ANI4IN responds with individual Report strings for all parameters.*
Get Model Number	
Command String: `< GET MODEL >`	
ANI4IN Response: `< REP MODEL {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 32 characters of the model number. The ANI4IN always responds with a 32 character model number.*

Get Serial Number	
Command String: `< GET SERIAL_NUM >`	
ANI4IN Response: `< REP SERIAL_NUM {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 32 characters of the serial number. The ANI4IN always responds with a 32 character serial number.*
Get Firmware Version	
Command String: `< GET FW_VER >`	
ANI4IN Response: `< REP FW_VER {yyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyy is 18 characters. The ANI4IN always responds with 18 characters.*
Get Audio IP Address	
Command String: `< GET IP_ADDR_NET_AUDIO_PRIMARY >`	
ANI4IN Response: `< REP IP_ADDR_NET_AUDIO_PRIMARY {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is a 15 digit IP address.*
Get Audio Subnet Address	
Command String: `< GET IP_SUBNET_NET_AUDIO_PRIMARY >`	
ANI4IN Response: `< REP IP_SUBNET_NET_AUDIO_PRIMARY {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is a 15 digit subnet address.*
Get Audio Gateway Address	
Command String: `< GET IP_GATEWAY_NET_AUDIO_PRIMARY >`	
ANI4IN Response: `< REP IP_GATEWAY_NET_AUDIO_PRIMARY {yyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyy is a 15 digit gateway address.*
Get Channel Name	

Command String: `< GET x CHAN_NAME >`	*Where x is ASCII channel number: 0 through 4.*
ANI4IN Response: `< REP x CHAN_NAME {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 31 characters of the channel name. The ANI4IN always responds with a 31 character name.*
Get Device ID	
Command String: `< GET DEVICE_ID >`	*The Device ID command does not contain the x channel character, as it is for the entire ANI4IN.*
ANI4IN Response: `< REP DEVICE_ID {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy is 31 characters of the device ID. The ANI4IN always responds with a 31 character device ID.*
Get Preset	
Command String: `< GET PRESET >`	
ANI4IN Response: `< REP PRESET nn >`	*Where nn is the preset number 01-10.*
Set Preset	
Command String: `< SET PRESET nn >`	*Where nn is the preset number 1-10. (Leading zero is optional when using the SET command).*
ANI4IN Response: `< REP PRESET nn >`	*Where nn is the preset number 01-10.*
Get Preset Name	

Command String:
`< GET PRESET1 >`
`< GET PRESET2 >`
`< GET PRESET3 >`
etc
	Send one of these commands to the ANI4IN.

ANI4IN Response:
 	
`< REP PRESET1 {yyyyyyyyyyyyyyyyyyyyyyyyy} >`
 	
`< REP PRESET2 {yyyyyyyyyyyyyyyyyyyyyyyyy} >`
 	
`< REP PRESET3 {yyyyyyyyyyyyyyyyyyyyyyyyy} >`
 	
etc
	Whereyyyyyyyyyyyyyyyyyyyyyyyyy is 25 characters of the preset name. The ANI4IN always responds with a 25 character preset name

Get Digital Audio Gain	
Command String: `< GET x AUDIO_GAIN_HI_RES >`	*Where x is ASCII channel number: 0 through 4.*
ANI4IN Response: `< REP x AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400. yyyy is in steps of one-tenth of a dB.*
Set Digital Audio Gain	
Command String: `< SET x AUDIO_GAIN_HI_RES yyyy >`	*Where x is ASCII channel number: 1 through 4. Where yyyy takes on the ASCII values of 0000 to 1400. yyyy is in steps of one-tenth of a dB.*
ANI4IN Response: `< REP x AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Increase Digital Audio Gain by n dB	
Command String: `< SET x AUDIO_GAIN_HI_RES INC nn >`	*Where x is ASCII channel number: 1 through 4. Where nn is the amount in one-tenth of a dB to increase the gain. nn can be single digit (n), double digit (nn), triple digit (nnn).*
ANI4IN Response: `< REP x AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1400.*
Decrease Digital Audio Gain by n dB	
Command String: `< SET x AUDIO_GAIN_HI_RES DEC nn >`	*Where x is ASCII channel number: 1 through 4. Where nn is the amount in one-tenth of a dB to decrease the gain. nn can be single digit (n), double digit (nn), triple digit (nnn).*
ANI4IN Response: `< REP x AUDIO_GAIN_HI_RES yyyy >`	*Where yyyy takes on the ASCII values of 0000 to 1280.*

Get Analog Audio Gain	
Command String: `< GET x AUDIO_GAIN >`	*Where x is ASCII channel number: 0 through 4.*
ANI4IN Response: `< REP x AUDIO_GAIN yy >`	*Where yy takes on the ASCII values of 00 to 51. yy is in steps of three dB.*
Set Analog Audio Gain	
Command String: `< SET x AUDIO_GAIN yy >`	*Where x is ASCII channel number: 1 through 4. Where yy takes on the ASCII values of 00 to 51. yy is in steps of three dB.*
ANI4IN Response: `< REP x AUDIO_GAIN yy >`	
Increment Analog Audio Gain	
Command String: `< SET x AUDIO_GAIN INC yy >`	*Where x is channel and takes on values 0, 1-4 (ANI4IN). Where yy is in 3 dB step. The resulting gain when the yy is applied is saturated to be in the range allowed in the SET.*
ANI4IN Response: `< REP x AUDIO_GAIN yy >`	*Where x is channel and takes on values 1-4 (ANI4IN). Where yy is in range of ANI4IN: 00 to +51 dB in 3 dB steps*
Decrement Analog Audio Gain	
Command String: `< SET x AUDIO_GAIN DEC yy >`	*Where x is channel and takes on values 0, 1-4 (ANI4IN). Where yy is in 3 dB step. The resulting gain when the yy is applied is saturated to be in the range allowed in the SET.*
ANI4IN Response: `< REP x AUDIO_GAIN yy >`	*Where x is channel and takes on values 1-4 (ANI4IN). Where yy is in range of ANI4IN: 00 to +51 dB in 3 dB steps*
Get Channel Audio Mute	
Command String: `< GET x AUDIO_MUTE >`	*Where x is ASCII channel number: 0 through 4.*
ANI4IN Response: `< REP x AUDIO_MUTE ON >`	*The ANI4IN will respond with one of these strings.*

`< REP x AUDIO_MUTE OFF >`	
Mute Channel Audio	
Command String: `< SET x AUDIO_MUTE ON >`	
ANI4IN Response: `< REP x AUDIO_MUTE ON >`	
Unmute Channel Audio	
Command String: `< SET x AUDIO_MUTE OFF >`	
ANI4IN Response: `< REP x AUDIO_MUTE OFF >`	
Toggle Channel Audio Mute	
Command String: `< SET x AUDIO_MUTE TOGGLE >`	

ANI4IN Response:
`< REP x AUDIO_MUTE ON >`
`< REP x AUDIO_MUTE OFF >`
	The ANI4IN will respond with one of these strings.

Flash Lights on ANI4IN	

Command String:
`< SET FLASH ON >`
`< SET FLASH OFF >`
	Send one of these commands to the ANI4IN. The flash automatically turns off after 30 seconds.

ANI4IN Response:
`< REP FLASH ON >`
`< REP FLASH OFF >`
	The ANI4IN will respond with one of these strings.

Turn Metering On	
Command String: `< SET METER_RATE sssss >`	*Where sssss is the metering speed in milliseconds. Setting sssss=0 turns metering off. Minimum setting is 100 milliseconds. Metering is off by default.*

ANI4IN Response:
 	
`< REP METER_RATE sssss >`
 	
`< SAMPLE aaa bbb ccc ddd >`
	Where aaa, bbb, etc is the value of the audio level received and is 000-060.
	
	aaa= output 1
	
	bbb= output 2
	
	ccc= output 3
	
	ddd= output 4

Stop Metering	
Command String: `< SET METER_RATE 0 >`	*A value of 00000 is also acceptable.*
ANI4IN Response: `< REP METER_RATE 00000 >`	
Get Sig/Clip LED	
Command String: `< GET x LED_COLOR_SIG_CLIP >`	*Where x is ASCII channel number: 0 through 4. It is not necessary to contin-ually send this command. The ANI4IN will send a REPORT message whenev-er the status changes.*

ANI4IN Response:
 	
`< REP x LED_COLOR_SIG_CLIP OFF >`
 	
`< REP x LED_COLOR_SIG_CLIP GREEN >`
 	
`< REP x LED_COLOR_SIG_CLIP AMBER >`
 	
`< REP x LED_COLOR_SIG_CLIP RED >`
	The ANI4IN will respond with one of these strings. This matches the sig/clip LEDs on the front of the ANI4IN.

Get LED Brightness	
Command String: `< GET LED_BRIGHTNESS >`	
ANI4IN Response: `< REP LED_BRIGHTNESS n >`	*Where n can take on the following val-ues:* 0 = LED disabled 1 = LED dim 2 = LED default
Set LED Brightness	

Command String: `< SET LED_BRIGHTNESS n >`	*Where n can take on the following values:* 0 = LED disabled 1 = LED dim 2 = LED default
ANI4IN Response: `< REP LED_BRIGHTNESS n >`	
Get Phantom Power Status	
Command String: `< GET x PHANTOM_PWR_ENABLE >`	

ANI4IN Response:
 	
`< REP x PHANTOM_PWR_ENABLE ON >`
 	
`< REP x PHANTOM_PWR_ENABLE OFF >`
	The ANI4IN will respond with one of these strings.

Turn on Phantom Power	
Command String: `< SET x PHANTOM_PWR_ENABLE ON >`	
ANI4IN Response: `< REP x PHANTOM_PWR_ENABLE ON >`	
Turn off Phantom Power	
Command String: `< SET x PHANTOM_PWR_ENABLE OFF >`	
ANI4IN Response: `< REP x PHANTOM_PWR_ENABLE OFF >`	
Get Mic Logic Switch Out	
Command String: `< GET x HW_GATING_LOGIC >`	*Where x is ASCII channel number: 0 through 4. It is not necessary to continually send this command. The ANI4IN will send a REPORT message whenever the status changes.*

ANI4IN Response:
`< REP x HW_GATING_LOGIC ON >`
`< REP x HW_GATING_LOGIC OFF >`
	The ANI4IN will respond with one of these strings.

Get Mic Logic LED In	
Command String: `< GET x CHAN_LED_IN_STATE >`	*Where x is ASCII channel number: 0 through 4.*

ANI4IN Response:
`< REP x CHAN_LED_IN_STATE ON >`
`< REP x CHAN_LED_IN_STATE OFF >`
	The ANI4IN will respond with one of these strings.

Set Mic Logic LED In	

Command String:
`< SET x CHAN_LED_IN_STATE ON >`
`< SET x CHAN_LED_IN_STATE OFF >`
	Send one of these commands to the ANI4IN.

ANI4IN Response:
`< REP x CHAN_LED_IN_STATE ON >`
`< REP x CHAN_LED_IN_STATE OFF >`
	The ANI4IN will respond with one of these strings.

Reboot ANI4IN (firmware > v2.0)	
Command String: `< SET REBOOT >`	
ANI4IN Response:	*The ANI4IN does not send a response for this command*
Get Error Events (firmware > v2.0)	
Command String: `< GET LAST_ERROR_EVENT >`	
ANI4IN Response: `< REP LAST_ERROR_EVENT {yyyyy} >`	*Where yyyy can be up to 128 charac-ters.*
Get Input Meter Mode (firmware > v2.0)	
Command String:	

`< GET INPUT_METER_MODE >`	

ANI4IN Response:
`< REP INPUT_METER_MODE PRE_FADER >`
`< REP INPUT_METER_MODE POST_FADER >`
	The ANI4IN will respond with one of these strings.

Set Input Meter Mode (firmware > v2.0)	

Command String:
`< SET INPUT_METER_MODE PRE_FADER >`
`< SET INPUT_METER_MODE POST_FADER >`
	Send one of these commands to the ANI4IN.

ANI4IN Response:
`< REP INPUT_METER_MODE PRE_FADER >`
`< REP INPUT_METER_MODE POST_FADER >`
	The ANI4IN will respond with one of these strings.

Get Limiter Engaged (firmware > v2.0)	
Command String: `< GET x LIMITER_ENGAGED >`	*Where x is ASCII channel number: 1 or 3. The limiter is only engaged when using summing mode*

ANI4IN Response:
`< REP x LIMITER_ENGAGED ON >`
`< REP x LIMITER_ENGAGED OFF >`
	The ANI4IN will respond with one of these strings.

Get Audio Summing Mode (firmware > v2.0)	
Command String: `< GET AUDIO_SUMMING_MODE >`	

ANI4IN Response:
`< REP AUDIO_SUMMING_MODE OFF >`
`< REP AUDIO_SUMMING_MODE 1+2 >`
`< REP AUDIO_SUMMING_MODE 3+4 >`
`< REP AUDIO_SUMMING_MODE 1+2/3+4 >`
`< REP AUDIO_SUMMING_MODE 1+2+3+4 >`
	The ANI4IN will respond with one of these strings.

Set Audio Summing Mode (firmware > v2.0)	
Command String: `< SET AUDIO_SUMMING_MODE OFF >`	*Send one of these commands to the ANI4IN.*

`< SET AUDIO_SUMMING_MODE 1+2 >`
 	
`< SET AUDIO_SUMMING_MODE 3+4 >`
 	
`< SET AUDIO_SUMMING_MODE 1+2/3+4 >`
 	
`< SET AUDIO_SUMMING_MODE 1+2+3+4 >`	

ANI4IN Response:
 	
`< REP AUDIO_SUMMING_MODE OFF >`
 	
`< REP AUDIO_SUMMING_MODE 1+2 >`
 	
`< REP AUDIO_SUMMING_MODE 3+4 >`
 	
`< REP AUDIO_SUMMING_MODE 1+2/3+4 >`
 	
`< REP AUDIO_SUMMING_MODE 1+2+3+4 >`
	The ANI4IN will respond with one of these strings.

Get RMS Audio Level (firmware > v2.0)	
Command String: `< GET x AUDIO_IN_RMS_LVL >`	*where x is channel number: 0: all channels ANI4IN: 1-4*
ANI4IN Response: `< REP x AUDIO_IN_RMS_LVLnnn >`	*where x is channel number defined in GET command. where nnn is audio level in the range of 000-060*
Get Peak Audio Level (firmware > v2.0)	
Command String: `< GET x AUDIO_IN_PEAK_LVL >`	*where x is channel number: 0: all channels ANI4IN: 1-4*
ANI4IN Response: `< REP x AUDIO_IN_PEAK_LVLnnn >`	*where x is channel number, defined in GET command. where nnn is audio level in the range of 000-060*
Get Network Audio Device Name	
Command String: `< GET NA_DEVICE_NAME >`	
ANI4IN Response: `< REP NA_DEVICE_NAME {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} >`	*Where {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} is a text string. Most devices allow device id to be up to 31characters. Value is padded with spaces as needed to ensure that 31 char are always reported.*
Get Network Audio Channel Name	
Command String:	*Where xx is channel number All channels: 0 ANI4OUT: 1-4*

`< GET NA_CHAN_NAME >`	
ANI4IN Response: `< REP xx NA_CHAN_NAME {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy}`	*Where xx is channel number. Where {yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy} is 31 char channel name. Value is padded with spaces as needed to ensure that 31 char are always reported.*
Get Control Network MAC Address	
Command String: `< GET CONTROL_MAC_ADDR >`	
ANI4IN Response: `< REP CONTROL_ MAC_ADDR yy:yy:yy:yy:yy:yy >`	*Where yy:yy:yy:yy:yy:yy is a 17 char literal string formatted as 6 octets, each separated by a colon. Example: 00:0E:DD:FF:F1:63*
Restore Default Settings (firmware > v2.0)	
Command String: `< SET DEFAULT_SETTINGS >`	*Request the device to set itself to default settings.*
ANI4IN Response: `< REP PRESET xx >`	*where xx = 00 if restore is successful*
Get LED State	
Command String: `< GET x LED_STATE_SIG_CLIP >`	*where x is channel number that takes on values: 0: all channels 1-4: individual channel*
ANI4IN Response: `< REP x LED_STATE_SIG_CLIP yyy > >`	*where x is channel number that takes on values: 1-4: individual channel; Where yyy is current LED state. Valid yyyvalues are: On - Steady, Flashing, Off*
Get PEQ Filter Enable (firmware > v2.0)	
Command String: `< GET xx PEQ yy >`	*Where xx is the PEQ block 01-04. Where yy is the PEQ filter 01-04 within the block. 00 can be used for all blocks or all filters.*
ANI4IN Response: `< REP xx PEQ yy ON >` `< REP xx PEQ yy OFF >`	

Set PEQ Filter Enable (firmware > v2.0)	

Command String:
`< SET xx PEQ yy ON >`
`< SET xx PEQ yy OFF >`
	Send one of these commands to the ANI4IN.

ANI4IN Response:
`< REP xx PEQ yy ON >`
`< REP xx PEQ yy OFF >`
	Where xx is the PEQ block 01-04. Where yy is the PEQ filter 01-04 within the block. 00 can be used for all blocks or all filters.

Get Encryption Status (firmware > v2.0)	
Command String: `< GET ENCRYPTION >`	*Get device level encryption status;*

ANI4IN Response:
`< REP ENCRYPTION ON >`
`< REP ENCRYPTION OFF >`
	Send one of these commands to the ANI4IN.