Revolabs Executive Elite™

Wireless Microphone System

Room Control API Guide

Models:

01-ELITEEXEC8	03-ELITEEXEC4-EU	03-ELITEEXEC8-JP
01-ELITEEXEC4	03-ELITEEXEC8-TW	03-ELITEEXEC4-JP
03-ELITEEXEC8-EU	03-ELITEEXEC4-TW	

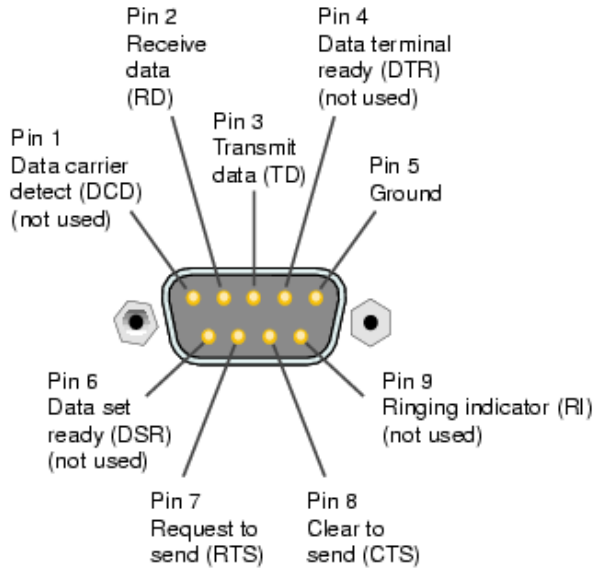
Contents

Using an Executive Elite with a Control System

The Executive Elite Base DSP can be monitored and controlled via an external control system. Communication can either be over the serial port of the Executive Elite, or over an IP connection. If several Executive Elite Base DSPs are part of an installation, each requires its own connection to the control system in order to communicate.

To use the control system API the type of external processor and the connection method has to be defined to match the external control system's settings. Open the local Web UI of the Executive Elite system. The External Control Configuration page is accessed by selecting External Control Configuration in the System Management menu. On this page, set the "Processor" field to Control System. Then define the Connection Mode that is being used – Telnet or Serial RS-232.

To use the Executive Elite with a RS-232 serial connection, a serial connection must be established between the DB9 serial port and an external control system. The baud rate configured in the Elite Web UI must match the baud rate of the external control system. The pin-out of the Executive Elite serial port is as follows:



To use an IP connection for the control system, with the Executive Elite being the server, the "Enable Telnet Server" switch on the System Configuration page of has to be set to "On".

Independent on whether a serial connection or an IP connection is being used, Executive Elite supports server mode or client mode for the control interface. Selecting the right mode is required for correct interoperability with the connected control system.

To setup the Executive Elite system in client mode using an IP connection, provide the IP address, port number, user name and password in the web UI to establish the

connection. Client mode in the executive Elite is persistent, meaning that if the server goes down or the connection is lost for any reason, the Executive Elite will periodically ping the server until the connection is re-established, without requiring a reboot.

If the Executive Elite is the server, only one device can login at a time. The login username is **revolabs** and the password is **elite**, both case sensitive. The port number is the standard Telnet port 23.

The Executive Elite can be configured for external mute on the External Control Configuration page of the local web UI. When external mute is set to On, the Executive Elite base DSP will always pass audio, even when the microphones are muted. In this case it is the responsibility of the installer to ensure that mute is performed in the external component of the installation.

Instead of a Room Control System, the Executive Elite system can receive commands also from a DSP system. In this case select the DSP from the list in the Processor drop-down menu.

NOTE: Both a control system and DSP cannot be connected to a Base Station as a serial control processor at the same time. When using a control system and DSP at the same time, DSP control is typically done from the room control system, with the Executive Elite either connected to the DSP or the Room Control System.

Serial Command & Return Strings

The Serial Strings of an Executive Elite Base Station begin with the Argument and terminates with a Carriage Return. The Command String structure is as follows:

[<Argument>] <Command> [ch <Channel Number> <Value>] <CR>

The argument is either 'set' or 'get'. While it is used on most commands, there are some command snot using the set and get argument. They are liste din the table below.

If specific microphone channels are affected by the command, the string 'ch' is followed by the microphone channel number, and the value which is sent ('set' command).

All variable changes on any microphone or the Executive Elite Base Station will result in a Return String. A Return String will begin with "val" and terminate with a Carriage Return. Return Strings will be received for every value change that has taken place in the microphones or base station including changes that are a result of a Command String sent to the Executive Elite Base Station, but also for changes initiated from the microphone or the web interface or front panel interface. The Return String structure is as follows:

val <Command> ch <Channel Number> <Value> <CR>

Channel Number:

The options for the <Channel Number> portion of the serial strings are 1-8 for an Executive Elite 8, and 1-4 for an Executive Elite 4 system, since each Base station can only control up to 8 or 4 microphones, respectively. The <Channel Number> corresponds to the channel number the microphone is paired to in the Executive Elite.

A <Channel Number> of "A" will control all microphones.

Examples:

The following string would set microphone channel 3 to be the chairman microphone for its mute group:

Command: *set chairmanmic ch 3 1* Return: *val chairmanmic ch 3 1*

The following strings would turn off all microphones on the Elite:

Command: *shutdownmic ch A* Return: *val shutdownmic ch A*

The following strings would retrieve the battery status of microphone 5 @ 98%:

Command: *get batterystatus ch 5* Return: *val batterystatus ch 5 98*

Legacy Command Strings

Legacy Commands that were used in the Executive HD have been maintained in the Executive Elite, to allow for Room Control programming using the Executive HD module to be used with Executive Elites. These commands are listed below in the legacy commands section of the tables.

The Command Interface of the Executive Elite has been expanded significantly with additional and new functionality. These new commands will not work with Revolabs Executive HD systems.

Command String Tables:

Property/Action	Command Format	Description	Response
addmictogroup	addmictogroup ch <channel number> <group name>	Add the given mic into the designated group.	addmictogroup ch <channel number> <group name>
antennafwversion	get antennafwversion	Retrieve antenna firmware version.	val antennafwversion <version>
antennalinkstatus	get antennalinkstatus	Retrieve the antenna link status, 0 - link down, 1 - link is up.	val antennalinkstatus <0/1>
antennamacaddr	get antennamacaddr	Retrieve antenna's MAC address.	val antennamacaddr <mac address>
audioquality	get audioquality	Retrieve the audio quality setting, 1 - SD, 2 - HD.	val audioquality <1/2>
audioquality	set audioquality <1/2>	Configure audio quality setting, 1 - SD, 2 - HD.	val audioquality <1/2>
audiosource	get audiosource	Retrieve back channel audio source.	val audiosource <analog/avb>
audiosource	set audiosource <analog/avb>	Configure back channel audio source.	val audiosource <analog/avb>
avbgptpmastermacaddr	get avbgptpmastermacaddr	Retrieve AVB ptp master mac address.	val avbgptpmastermacaddr <mac address>
avbgptpmode	get avbgptpmode	Retrieve AVB ptp mode.	val avbgptpmode <mode>
avblinkstatus	get avblinkstatus	Retrieve AVB link status.	val avblinkstatus <status>
avbmacaddr	get avbmacaddr	Retrieve AVB mic address.	val avbmacaddr <mac address>
backchannel	get backchannel	Retrieve the back channel setting, 0 - off, 1 - on.	val backchannel <0/1>
backchannel	set backchannel <0/1>	Configure back channel setting, 0 - off, 1 - on.	val backchannel <0/1>
batterystatus	get batterystatus ch <channel number>	Retrieve the given microphone's battery status.	get batterystatus ch <channel number> <percentage>
chairmanmic	get chairmanmic ch <channel number>	Retrieve the given microphone's chairman state, For response, 0 - non-chairman, 1 - chairman.	val chairmanmic ch <channel number> <0/1>
chairmanmic	set chairmanmic ch <channel number> <0/1>	Configure the given microphone's chairman state, 0 - non-chairman, 1 - chairman.	val chairmanmic ch <channel number> <0/1>
chairmanmicgroup	set chairmanmicgroup ch <channel number> <group name>	Configure a microphone in a certain group to be chairman mic.	val chairmanmicgroup ch <channel number> <group name>
chairmanmicgroup	get chairmanmicgroup <group name>	Get current chairman mic in the given group.	val chairmanmicgroup <group name> <mic id>
chairmanmute	get chairmanmute ch <channel number>	Retrieve the given channel's chairman mute state. 0 - all unmuted, 1 – members muted, 2 - all muted	val chairmanmute ch <channel number> <0/1>
chairmanmute	set chairmanmute ch <channel number> <0/1/2>	Configure the given channel's chairman mute state. 0 - all unmuted, 1 – members muted, 2 - all muted	val chairmanmute ch <channel number> <0/1/2>
cloudaccountname	get cloudaccountname	Retrieve cloud account name.	val cloudaccountname <account name>
clusterhostlist	get clusterhostlist	Retrieve the list of host IP addresses currently clustered.	val clusterhostlist <IP 1> <IP 2> ..
clusterid	get clusterid	Retrieve the cluster id.	val clusterid <id>
clusterid	set clusterid <id>	Configure the cluster id.	val clusterid <id>
countryregion	get countryregion	Retrieve system's country/region setting. 1 - USA/Canada, 3 - China, 4 - Europe, 5 - UK, 6 - Japan, 7 - Australia, 9 - Taiwan, 10 - Mexico, 11 - Argentina, 12 - Chile, 13 - Brazil, 14 - Singapore, 16 - Hong Kong, 17 - Indonesia, 21 - Malaysia, 22 - South Africa,	val countryregion

Property/Action	Command Format	Description	Response
		23 - Costa Rica, 24 - Venezuela, 25 - Middle East	
create mutegroup	create mutegroup <new group name>	Create a new mute group.	create mutegroup <new group name>
currentipaddress	get currentipaddress	Retrieve current IP address.	val currentipaddress <ip address>
currentipdefaultgateway	get currentipdefaultgateway	Retrieve current default gateway address.	val currentipdefaultgateway <ip gateway>
currentipdns	get currentipdns	Retrieve current IP DNS address.	val currentipdns <ip dns>
currentipsubnet	get currentipsubnet	Retrieve current IP subnet.	val currentipsubnet <ip subnet>
delete mutegroup	delete mutegroup <group name>	Delete an existing group.	delete mutegroup <group name>
deviceid	get deviceid ch <channel number>	Retrieve microhone's dect id.	val deviceid ch <channel number> <dect id>
echo	set echo <0/1>	Echo the entered command. 0 - off, 1 - on.	<no response>
enableonlineaccess	get enableonlineaccess	Retrieve online access enable/disabled state, 0 - disable, 1 - enable.	val enableonlineaccess <0/1>
enableonlineaccess	set enableonlineaccess <0/1>	Configure enable/disable online access. 0 - disable, 1 - enable.	val enableonlineaccess <0/1>
eq	get eq ch <channel number>	Retrieve the four EQ values associated with the given channel.	get eq ch <channel number> <eq1> <eq2> <eq3> <eq4>
eq	set eq ch <channel number> <eq1> <eq2> <eq3> <eq4>	Configure the four EQ values associated with the given channel, EQ range -10 to 10 db.	val eq ch <channel number> <eq1> <eq2> <eq3> <eq4>
groupbyindex	get groupbyindex <index number>	Retrieve the group name by index	val groupbyindex <channel number> <group name>
groupcount	get groupcount	Retrieve the group count.	val groupcount <group count>
groupmutetype	get groupmutetype <group name>	Retrieve group mute type, 0 - Individual mute, 1 - Tabletop mute, 2 - Push to talk.	val groupmutetype <group name> <0/1/2>
groupmutetype	set groupmutetype <group name> <0/1/2>	Configure group mute type, 0 - Individual mute, 1 - Tabletop mute, 2 - Push to talk.	val groupmutetype <group name> <0/1/2>
groupstandbymode	get groupstandbymode <groupname>	Retrieve group's standby mode state.	val groupstandbymode <groupname> <0/1>
groupstandbymode	set groupstandbymode <groupname> <0/1/2>	Set group standby mode, 1 - Enter standby, 0 - leave standby, 2 - toggle.	val groupstandbymode default 1
hipassfilter	get hipassfilter ch <chanel number>	Retrieve the high pass filter value configured for given channel.	val hipassfilter ch <chanel number> <0/110/140/175/220>
hipassfilter	set hipassfilter ch <channel number> <0/110/140/175/220>	Configure the high pass filter value for a given channel.	val hipassfilter ch <chanel number> <0/110/140/175/220>
ipdhcp	get ipdhcp	Retrieve the DHCP enable/disable state. 0 - disabled, 1 - enabled.	val ipdhcp <0/1>
ipntp	get ipntp	Retrieve the currently configured NTP servers.	get ipntp <server1> <server2> …
language	get language	Retrieve the language code. EN - English, FR - French, GE – German	val language <language code>
language	set language <EN/FR/GE>	Configure the language code. EN - English, FR - French, GE – German	val language <language code>
legacymode	get legacymode	Retrieve the legacy mode setting value. 0 – Legacy mode off, 1 - Legacy mode on	val legacymode <0/1>
legacymode	set legacymode <0/1>	Configure the legacy mode. 0 – Legacy mode off, 1 - Legacy mode on	val legacymode <0/1>
logmask	get logmask	Retrieve the debug log level, minimum severity, 0 - debug, 1 - warning, 2 - error, 3 - off.	val logmask <0/1/2/3>

Property/Action	Command Format	Description	Response
logmask	set logmask <0/1/2/3>	Configure the debug log level, minimum severity, 0 - debug, 1 - warning, 2 - error, 3 - off.	val logmask <0/1/2/3>
lopassfilter	set lopassfilter ch <channel number> <0/12000/8000/4000>	Configure the low pass filter value for a given channel.	val lopassfilter ch <chanel number> <0/12000/8000/4000>
lopassfilter	get lopassfilter ch <chanel number>	Retrieve the low pass filter value configured for given channel.	val lopassfilter ch <chanel number> <0/12000/8000/4000>
macaddr	get macaddr	Retrieve system's MAC address.	val macaddr <mac address>
mastermutemicgroup	get mastermutemicgroup <group name>	Retrieve the given group's master mute state, 0 - unmute, 1 - muted.	val mastermutemicgroup <group name> <0/1>
mastermutemicgroup	set mastermutemicgroup <group name> <0/1>	Configure the given group's master mute state, 0 - unmute, 1 - muted.	val mastermutemicgroup <group name> <0/1>
rename mutegroup	rename mutegroup <source group> <target group>	Rename the given source group.	rename mutegroup <source group> <target group>
micgain	set micgain ch <channel number> <gain value>	Set gain value in db, channel number 1-8 or A.	val micgain ch <channel number> <gain value>
micgain	get micgain ch <channel number>	Get gain value in db, channel number 1-8 or A.	val micgain ch <channel number> <gain value>
micgroup	get micgroup ch <channel number>	Retrieve the group name to which the given mic belongs.	val micgroup ch <channel number> <group name>
micstandbymode	get micstandbymode ch <channel number>	Retrieve mic standby status, channel number 1-8 or A	val micstandbymode ch <channel number> <0/1>
micstandbymode	set micstandbymode ch <channel number> <0/1/2>	Set mic into standby mode, 1 - Enter standby, 0 - leave standby, 2 - toggle.	val micstandbymode ch <channel number> <0/1>
micstatus	get micstatus ch <channel number>	Retrieve the given microphone's current status, 0 - off, 1 - on, 2 - standby, 3 - charging, 4 - out of range, 5 - not paired, 6 - pairing, 7 - unpairing, 8 – updating	val micstatus ch <channel number> <status>
mictype	get mictype ch <channel number>	Retrieve the given's microphone's type. 0 - Wearable, 1 - Omni, 2 - Directional, 3 - XLR, 4 - Mini XLR, 5 - Gooseneck, 10 - Handheld, 11 – unknown	val mictype ch <channel number> <type>
micversion	get micversion ch <channel number>	Retrieve mic software version.	val micversion <channel number> <version number>
mutestatus	get mutestatus ch <channel number>	Retrieve the mic mute status, value is a comma delimited string: <mute>,<mute type>,<mute lock>	val mutestatus ch <channel number> <mute>,<mute type>,<mute lock>
onlineregstate	get onlineregstate	Retrieve online registration state, 0 - unregistered, 1 -registered, 2 - Do not remind me.	val onlineregstate <state>
regnotify	Regnotify	Register to listen to system notifications.	regnotify
outputgain	get outputgain ch <channel number>	Retrieve the given channel's output gain.	val outputgain ch <channel number> <output gain>
outputgain	set outputgain ch <channel number> <output gain>	Configure the given channel's output gain, output gain range -20 to 5 db	val outputgain ch <channel number> <output gain>
outputlevel	get outputlevel ch <channel number>	Retrieve the channel output level, channel number 1/2/../8/A, value 0 - mic level, 1 - line level.	val outputlevel ch <channel number> <0/1>
outputlevel	set outputlevel ch <channel> <value>	Configure the given channel's output level, channel number 1/2/…/8/A, value 0 – mic level, 1 – line level	val outputlevel ch <channel number> <0/1>
preampgain	get preampgain ch <channel number>	Retrieve the given channel's preamp gain.	val preampgain ch <channel number> <preamp gain>
preampgain	set preampgain ch <channel number> <preamp gain>	Configure the given channel's preamp gain, preamp gain range -20 to 0 db.	val preampgain ch <channel number> <preamp gain>
productname	get productname	Retrieve the product name.	val productname <product name>

Property/Action	Command Format	Description	Response
pushtotalk	get pushtotalk <group name>	Retrieve group push to talk setting value, 1 - push to talk set, 0 - not set.	val pushtotalk <group name> <0/1>
pushtotalk	set pushtotalk <group name> <0/1/2>	Configure group push to talk setting, 1 - push to talk set, 0 - not set, 2 - toggle.	val pushtotalk <group name> <0/1/2>
pushtotalkmic	get pushtotalkmic ch <channel number>	Retrieve microphone's push to talk state. 1 - push to talk set, 0 - not set.	val pushtotalkmic ch <channel> <0/1>
regnotify	regnotify	Enables system notification	
removemicfromgroup	removemicfromgroup ch <channel number> <group name>	Remove the given mic from the designated group.	removemicfromgroup ch <channel number> <group name>
restart	set restart ch <channel number>	Restart the given channel, if channel A is specified, restart all mics.	val restart ch <channel number>
restart	set restart <component>	Restart the specified component, all - all devices, AR - antenna receiver, base - base receiver.	val restart <component>
rfautoscaling	get rfautoscaling	Retrieve the RF auto scaling setting value, 0 - off, 1 - on.	val rfautoscaling <0/1>
rfautoscaling	set rfautoscaling <0/1>	Configure the RF auto scaling setting, 0 - off, 1 - on.	val rfautoscaling <0/1>
rfpowerlevel	get rfpowerlevel	Retrieve RF power level. 2 = (0 - 20 ft), 3 = (10 - 40 ft), 4 = (30 - 75 ft), 5 = (50 -150 ft), 6 = (100 - 300 ft)	val rfpowerlevel <2/3/…/6>
rfpowerlevel	set rfpowerlevel <2/3/…/6>	Configure RF power level. 2 = (0 - 20 ft), 3 = (10 - 40 ft), 4 = (30 - 75 ft), 5 = (50 -150 ft), 6 = (100 - 300 ft)	val rfpowerlevel <2/3/…/6>
serialnumber	get serialnumber <base/antenna>	Retrieve component serial number.	val serialnumber base <serial number>
shutdownmic	shutdownmic ch <channel number>	Power the microphone off.	shutdownmic ch <channel number>
startupunmutegroup	get startupunmutegroup <group name>	Retrieve given group's unmute at startup state, 0 - mute at startup, 1 - unmute at startup.	val startupunmutegroup <group name> <0/1>
startupunmutegroup	set startupunmutegroup <group name> <0/1>	Configure given group's unmute at startup state, 0 - mute at startup, 1 - unmute at startup.	val startupunmutegroup <group name> <0/1>
systemclockmode	get systemclockmode	Retrieve system clock mode, 0 - Internal, 1 - External, 2 - Auto.	val systemclockmode <0/1/2>
systemclockmode	set systemclockmode <0/1/2>	Configure system clock mode, 0 - Internal, 1 - External, 2 - Auto.	val systemclockmode <0/1/2>
systemclocksource	get systemclocksource	Retrive system clock source, 0 - local, 1 - Sync bus, 2 - Avb.	val systemclocksource <0/1/2>
systemclocksource	set systemclocksource <0/1/2>	Configure system clock source, 0 - local, 1 - Sync bus, 2 - Avb.	val systemclocksource <0/1/2>
systemclockstatus	get systemclockstatus	Retrieve system clock status. 0 - no system clock, 1- system clock	val systemclockstatus <0/1>
systemname	get systemname	Retrieve system name.	val systemname <name>
tabletopmutegroup	get tabletopmutegroup <group name>	Retrieve the given group's tabletop mute state, 0 - not tabletop mute, 1 - tabletop mute.	val tabletopmutegroup <group name> <0/1>
tabletopmutegroup	set tabletopmutegroup <group name> <0/1>	Configure the given group's tabletop mute state, 0 - not tabletop mute, 1 - tabletop mute.	val tabletopmutegroup <group name> <0/1>
usbipaddr	get usbipaddr	Retrieve the USB ethernet interface IP address.	val usbipaddr <ip address>
version	get version	Retrieve the system bundle version.	get version <version>

The command 'regnotify' listed above will turn on notifications. The following table lists the notifications sent by the system.

Notifications	Notification Format	Description
mute_group_change	notify mute_group_change <group name> <action>	Notify client that changes that affect a given group have occurred. Actions include 'add' - add, 'del' - delete, 'ren' - rename.
mute_group_member_change	notify mute_group_member_change <group name> bid <box id> active <0/1> type <type> index <mic index>	Notify client that a mic has changed its group.
mute_group_mute_type_change	notify mute_group_mute_type_change <group name> <mute type>	Notify client that the given mute group's mute type has changed, 0 - Individual mute, 1 - Tabletop mute, 2 - Push to talk.
property_change_audio_quality	notify property_change_audio_quality <1/2>	Notify client that the audio quality setting has changed, 1 - SD, 2 - HD.
property_change_back_channel_audio	notify property_change_back_channel_audio <0/1>	Notify client that the audio back channel setting is changed, 0 - no back channel, 1 - back channel on.
property_change_eq_<eqX>	notify property_change_eq_<eqX> ch<channel number> <value>	Notify client the given channel's eq value changed, <eqX> - X can be 1/2/3/4, and <value> will range from -10 db to +10 db.
property_change_gain	notify property_change_gain ch<channel number> <gain>	Notify client the given microphone's input gain changed, gain value range from -20 to 0 db.
property_change_hipass_filter	notify property_change_hipass_filter ch<channel number> <frequency>	Notify client the given channel's high pass filter value changed,possible frequency values include 0, 110, 140, 175 and 220 Hz.
property_change_legacy_mode	notify property_change_legacy_mode <0/1>	Notify client that the legacy mode has changed, 0 - legacy off, 1 - legacy on.
property_change_lock	notify property_change_lock ch<channel number> <0/1>	Notify client the given microphone's mute lock changed, 0 - unlocked, 1 - locked.
property_change_lowpass_filter	notify property_change_lowpass_filter ch<channel number> <frequency>	Notify client the given channel's low pass filter value changed, possible frequency values include 0, 12000, 8000, and 4000 Hz.
property_change_output_level	notify property_change_output_level ch<channel number> <0/1>	Notify client the given channel's output level changed, 0 - mic, 1 - line.
property_change_output_gain	notify property_change_output_gain ch1 -4	Notify client the given channel's output gain changed, gain value range from -20 db to 5db.
property_change_rf_auto_scaling	notify property_change_rf_auto_scaling <0/1>	Notify client that the RF auto scaling setting is changed, 0 - off, 1 - on.
property_change_rf_power	notify property_change_rf_power <value>	Notify client the RF power setting is changed, 2 = (0 - 20 ft), 3 = (10 - 40 ft), 4 = (30 - 75 ft), 5 = (50 -150 ft), 6 = (100 - 300 ft)
property_change_system_name	notify property_change_system_name <new name>	Notify client that the system name is changed.
set_mute_status	notify set_mute_status ch<channel number> <mute>,<mute type>,<mute lock>	Notify client the given microphone's mute status changed. <mute> values: 0 - unmuted, 1 - muted <mute type> values: 0 - Individual Mute, 1 - Tabletop mute, 2 - Push to talk mute <mute lock> values: 0 - unlocked, 1 - locked
startup_unmute_group	notify startup_unmute_group <group name> <state>	Notify client that the given mute group's unmute at startup setting changed, 1 - unmute at startup, 0 - mute at startup.

Notifications	Notification Format	Description
status_change_antenna_link	notify status_change_antenna_link <status>	Notify client antenna link status changed, 0 - unlinked, 1 - linked.
status_change_battery	notify status_change_battery ch<channel number> <percentage>	Notify client the given microphone's battery status changes.
status_change_mic	notify status_change_mic ch<channel number> <status>	Notify client the given microphone's status changed, 0 - off, 1 - on, 2 - standby, 3 - charging, 4 - out of range, 5 - not paired, 6 - pairing, 7 - unpairing, 8 - updating
status_group_mute	notify status_group_mute <group name> <group mute state>	Notify client the given group's mute status changed. 8 - group muted, 0 - group unmuted.
upgrade_status_change	notify upgrade_status_change <status>	Notify client system upgrade status changed.
upgrade_status_change	notify upgrade_status_change <component> <status>	Notify client components upgrade status changed.

Legacy Command Tables:

Property/Action	Command Format	Description	Response
batt	get batt ch <channel number>	Retrieve battery status for mic in specified channel.	val batt ch <channel number><1/2/3/4>
fp	set fp <0/1>	Set front panel display lock on or off; 0 - Off, 1 – On	val fp <0/1>
fw	get fw	Retrieve system firmware version.	val fw <0/1>
gain	set gain ch <channel number> <0/1/2/80-120>	Configure input gain, values 80 - 100 set the gain from -20 to 0 db values 101 – 120 will all set the gain 0 db 1 - modify gain by +1, 2 - modify gain by -1, 0 - set gain to 0	val gain ch <channel number> <0/1/2/80-120>
gain	get gain ch <channel number>	Retrieve input gain for specified channel.	val gain ch <channel number><80-120>
lock	set lock ch <channel number> <0/1>	Turns Mic Lock on or off; 0 - Off, 1 - On	val lock ch <channel number> <0/1>
lock	get lock ch <channel number>	Retrieve Mic Lock for specified channel.	val lock ch <channel number> <0/1>
mute	set mute ch <channel number> <0/1>	Mute or unmute specified channel: 0 - Unmute, 1 – Mute	val mute ch <channel number> <0/1>
mute	get mute ch <channel number>	Retrieve mute state for specified channel.	val mute ch <channel number><0/1/2/3/4/5>
pair	set pair <0/1>	Set base pairing lock on or off; 0 - Off, 1 – On	val pair <0/1>
pair	set pair ch <channel number> <0/1>	Activate pairing mode for selected channel. 0 - Inactive, 1 – Active	val pair ch <channel number> <0/1>
pair	get pair ch <channel number>	Retrieve base pairing status for specified channel.	val pair ch <channel number><0/1>
pwr	set pwr <channel number>	Power off microphone for selected channel.	val pwr <channel number>
um	set um <0/1>	Set unmute at startup; 0 - Off, 1 – On	val um <0/1>
tm	set tm <0/1>	Set tabletop mute on or off; 0 - Off, 1 – On	val tm <0/1>
type	get type ch <channel number>	Retrieve mic type for specified channel.	val type ch <channel number><1/2/3/4/5>
um	get um	Retrieve unmute at startup state.	val um <0/1>
tm	get tm	Retrieve tabletop mute state.	val tm <0/1>

Legacy Notifications	Notification Format	Description
batt	val batt	Mic battery status. 1 - 25%, 2 - 50%, 3 - 75%, 4 - 100%
gain	val gain…	Reports gain value 80 - 100 = gain values -20db - 0db, in 1db increments. (ex: 81 = -19db) 101 – 120 = gain value 0db
lock	val lock…	Mic lock status.
mute	val mute…	Mic mute state; 0 - unmuted, 1 - self muted, 2 - muted, 3 - microphone off, 4 - microphone out of range, 5 - microphone charging
pair	val pair…	Channel pairing status; 0 - Inactive, 1 – Active
type	val type…	Mic type; 1 - Lapel (Wearable), 2 - Omnidirrectional Tabletop, 3 - Directional Tabletop, 4 - XLR Adapter, 5 - Countryman Adapter