VLX Series Protocol Guide

Document Version Date: 2019-0214

Minimum version requirement- CR 3.22.0

Table of Contents

1 About this guide

This document uses the following conventions

1.1 Symbols

{ } Used to indicate required parameters in the syntax.

*Ex. Command **{**Parameter**}** means Parameter is required*

[ ] Used to indicate optional parameters in the syntax.

*Ex: Command **[**Parameter**]** means Parameter is optional*

| Used to indicate exclusive parameters, used between the parameters.

*Ex. Command "option1" **|** "option2" means that either "option1" or "option2" can be used, but not both at the same time.*

" " Quotes are used to indicate text string input

*Ex. Command **"**text**"** means text is the input string.*

__ Underscore is used to indicate number input.

*Ex. Command **number** means number is replaced with hex or decimal number.*

2 Introduction

This document describes everything a programmer needs to control the VLX Series using custom "control application software" (here after referred as "application SW), using the provided API commands. This will allow 3rd party control to take advantage of all the features the VLX Series offers.

The Application SW can directly connect to the VLX series devices on the network. Using the discovery protocol described below, the Application SW can discover all devices (device_id IP address etc.) on the same network.

2.1 Device Discovery Protocol

On power on, all the VLX devices will be listening to UDP port 3333 of multicast IP 225.1.0.0
When a matching UDP query (of the following type) is received on the UDP port 3333 of this multicast IP, the device will respond to UDP port 3334.

The above scheme can be used by the application software to detect all VLX devices on the network. The application software can open UDP port 3334 and while listening to this port, send out the UDP query shown below to multicast IP 225.1.0.0 at UDP port 3333.

2.1.1 UDP request for discovery

The format of the UDP request is as follows:

```
typedef struct _query_struct_
{
    AST_Device_Type device_type;
    AST_Device_Function device_function;
}query_struct
```

device type can have values as follows:

- 0: for all devices
- 1: for all “encoder type devices
- 2: for all decoder type devices

The format of device type is as follows

```
typedef enum _AST_Device_Type_
{
        Type_Any = 0,
        Type_encoder,
        Type_decoder,
        Type_Unknown,
 } AST_Device_Type ;
```

Device function should be 0 (other values are for future updates)

2.1.2 Response for UDP request (discovery)

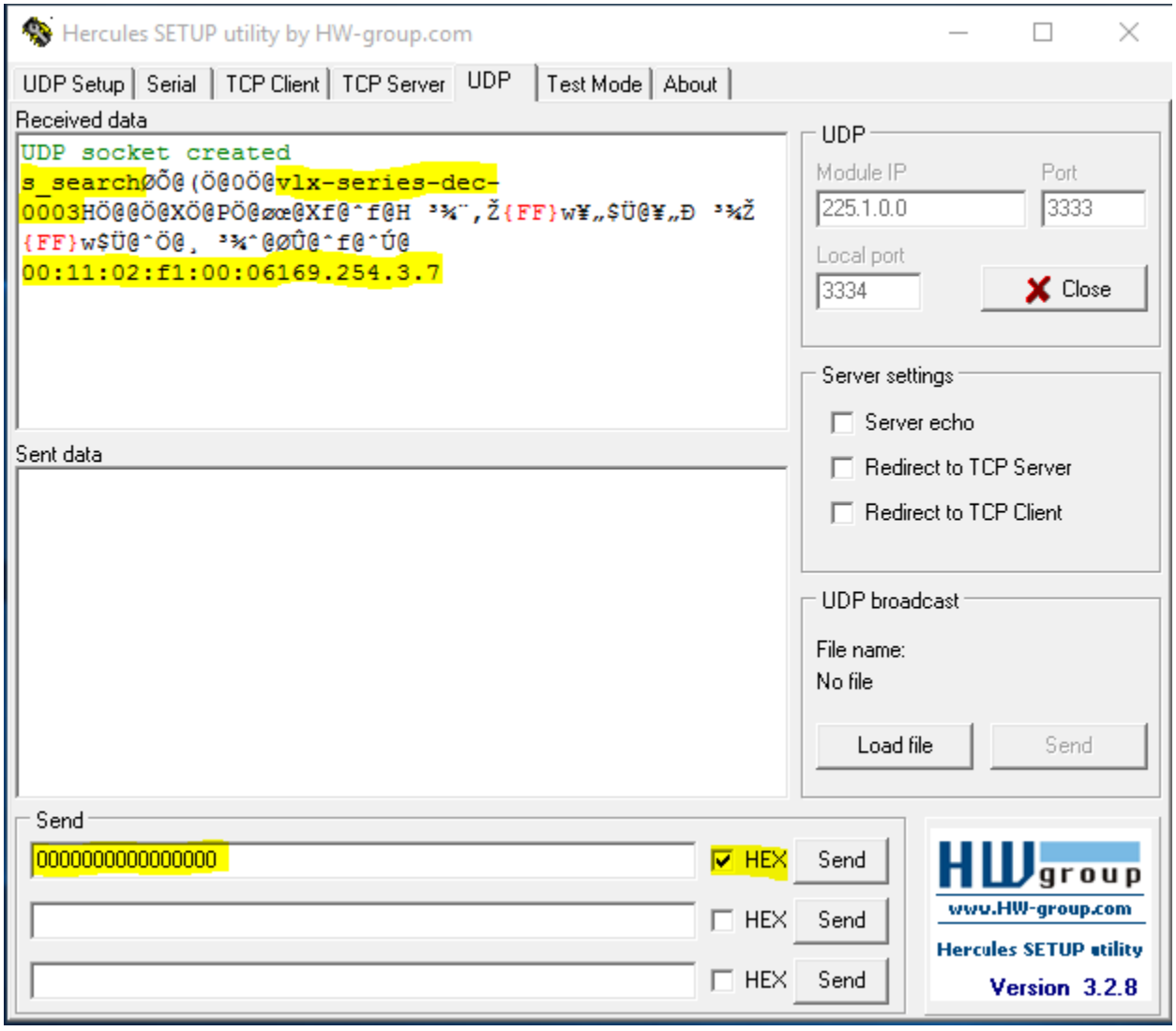
The response structure from individual VLX units is defined below.

```
typedef struct _reply_struct_
{
        AST_Device_Type device_type;
        AST_Device_Function device_function;
        char device_status[MAX_STATUS_LENGTH];
        char device_name[MAX_NAME_LENGTH];
        char device_macid[MAX_MACID_LENGTH];
        char device_ip[MAX_IP_LENGTH];
}reply_struct
```

Device type and device functions have the same meaning as described above.
Device_id is the hostname which identifies the vlx-series device in the network.

Usage

Try sending sixteen '0's(8 bytes) as hex via UDP to 225.1.0.0 port 3333 and listen on port 3334 in your PC.



2.2 Connection and API Commands

Once devices are “discovered” the application SW can connect to each of the VLX series device and issue commands using the Telnet protocol. This allows any Telnet client to be used to configure the system.
By default, VLX device listens on **TCP port 6970 for telnet commands**.

API commands are single-line text commands. For each command, the VLX device responds with a JSON response which contains the return status (i.e. whether the command succeeded or not) and, if successful, the return value of the command.

Programmers may refer to the following RFC specifications for information on how the Telnet protocol works:

Title	Internet Engineering Task Force (IETF) links
Telnet Protocol Specification	http://tools.ietf.org/html/rfc854
Telnet Option Specifications	http://tools.ietf.org/html/rfc855
The JavaScript Object Notation (JSON) Data Interchange Format	http://tools.ietf.org/html/rfc7159

More details about the JSON response format are in section 3 "Commands overview".

2.3 Application Compatibility and API Version Numbering Scheme

Starting with API version 0.1.0, the API uses a version numbering scheme which express compatibility information. A version number has the format ***major.minor.revision***.

Revisions introduce fixes or internal improvements that do not affect the programming interface. Versions that differ in their ***minor*** number are ***backward compatible***, but not forward compatible, because versions with a new minor number introduce new commands or parameters, new JSON fields, or other new functionality not present in earlier versions.

To ensure compatibility of your application with future API versions, please follow these simple rules:

- As new API versions will add new fields to the JSON response returned by commands, it is important that your application ignore fields it does not know about.
- As new API versions will add new node, subscription and stream types, it is important that your application ignore node, subscription and stream types it does not know about

The first thing an application is expected to do right after connecting to the VLX device is to get the VLX FW version number (using get api version command) and check that the version number is compatible with the application.

2.4 Immediate versus Background Commands

There are two types of commands in the VLX Series API:

- Commands that return with either **SUCCESS** or **ERROR** status & executes immediately (Immediate commands)
- Commands that return **PROCESSING** status and initiate execution in the background, which takes longer time. (Background commands).

The lifecycle of a background command is as follows:

- As the background command gets executed the device “status” gets updated. So the application SW can poll the device status periodically and keep track of the execution status of the background commands. Background commands in the current version are “join” and “start”

- For eg. if a “join” command is issued to VLX (decoder) device (to subscribe to a video stream from an encoder in the network) the device will respond with “PROCESSING” and start getting connected to the specified encoder stream. The stream connection status will be reflected in the devices “state”, which can be polled by the application SW.

2.5 Device Identity (Device_ID)

Every device in a network should have a unique identity. For VLX devices, the device ID is formed by combining following two parameters kept in the respective device.

- **Device name:** a string of max 20 alphanumeric characters including “-“. Default device name will be “vlx-series-enc-” for an encoder and “vlx-series-dec-“ for a decoder. This can later be changed partially by using “set hostname_prefix” command. ‘–dec-‘ or ‘-enc-‘ part of the device name changes automatically depending on whether the device is configured as encoder or decoder. This part cannot be changed using “set hostname_prefix” command.
- **Device number:** a four-digit decimal number. This should be unique for every device in a network. Default (factory set) device number will be based on its serial number. This can later be changed by using “set hostname_id” command**. Care should be taken to ensure no duplication of device numbers in a network.**

Example: vlx-series-enc-1234

2.6 Streams and Subscriptions

Control of data routing between devices uses a stream and subscription abstraction:

- *Streams* are the *sending endpoints* while
- *Subscriptions* are the *receiving endpoints*.

Each stream or subscription has a type, and only streams and subscriptions of the same type can be associated together. Streams are either joinable or switchable, depending on their type.

Joinable streams are configured to *send data to a multicast channel* using the **start** command, and subscriptions of receivers which are to receive that data are configured to join that channel using the **join** command. A subscription can leave a multicast channel either by joining another one or with the **leave** command.

Switchable streams are configured to directly send data to a target device's subscription, or to the subscriptions of a group of devices or to the API host using the **switch** command. No configuration of the receiving device is necessary for switchable stream types.

Both joinable and switchable streams can be configured to stop transmitting using the **stop** command.

Note: All encoder devices, on power on, check for the video input on the selected video IN port for streaming video, and if found a source, it will start video stream to the multicast (stream) IP address unique to the device. This enables any decoder devices in the network to “join” the stream as required.

Encoder devices join specific video stream based on the “join” command received over the API.

2.6.1 Multicast IP Address Management

Every VLX Device (configured as Encoder) assigns a unique multicast IP address to its streams. Each active joinable stream uses this multicast IP address.

Note: The multicast IP address is formed based on the last 4 digits of the “device_name”. for example, if device name of an encoder vlx-series-enc-1234, its multicast IP address for Video stream will be 225.0.112.34. So, in one network, no two encoders should have same “device number” field in the “device_id”.

By default, the audio and video from the selected video IN port (HDMI 1 or 2) will be streamed together. If required to combine video from Video IN source and audio from the analog audio IN, this can be done by enabling **mode split_hdmi_audio feature**. In this case, audio stream will have a different multicast IP address.

3 Commands overview

All commands should be separated by a delimeter. The applicable delimiters are \r,\n or \r\n (0x0A,0x0D)

Note: When using clients like TeraTerm these delimiters are automatically inserted on pressing ENTER Key.

All commands generate a JSON response like one of the following:

```
{
        "status":”String”,
        "result":”Varies”,
        "error":”String”
}
or
{
        "status":”String”,
        "error":”String”
}
```

Field name	Type	Description
Status	String	The resulting status which is one of the following: • **SUCCESS**: The command succeeded. • **ERROR**: The command failed. • **PROCESSING:** Background command getting executed • **BUSY:** applicable in bitmap commands when some other command execution is already in progress.
Result	Varies	Return value of the command. Set when the status is **SUCCESS**, null otherwise. This varies for each command. The return value is documented in the command reference section of each individual command.
Error	String	Error message describing the reason why the command failed. Set when the status is **ERROR**, null otherwise.

An error message can be any one of the following strings

- **INVALID PARAMETER:** An invalid argument was specified for the command.
- **INVALID PARAMETER COUNT:** Number parameters not correct.
- **INVALID_COMMAND:** The entered command (or subcommand) does not exist.
- **INVALID_STATE:** The specified command cannot be executed given the current state of the system. The same command may succeed later if the condition is resolved.
- **IO_ERROR**: The operation failed due to an input/output error (for example, disk read/write failure).
- **UNSUPPORTED_FEATURE:** The requested feature/module is not supported.

4 Core Command Reference

4.1 Version

4.1.1 Command usage

version

4.1.2 Description

Get version number of the device API and firmware.

4.1.3 Return Value

```
{
        "status":“string”,
        "api_version":“string”,
        "fw_version":“string”,
        "error":“string”
}
```

4.1.4 Example

version

```
{
        "status" : "SUCCESS",
        "api_version":"0.1.0",
        "fw_version":"1.0.4",
        "error":“NULL”
}
```

4.2 Factory_default

4.2.1 Command usage

factory_default

4.2.2 Description

Restore the settings of device to factory default.

4.2.3 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.2.4 Example

factory_default

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.3 Debug

4.3.1 Command usage

debug {on} or **debug {off}**

4.3.2 Description

Debug console of VLX device can be enabled and disabled. This command returns error status if SoIP mode is ON since the debug console will be in disabled mode by default in SoIP mode.

4.3.3 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.3.4 Example

debug on

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

debug off

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

If SoIP mode is enabled

debug on

```
{
        "status":“ERROR”,
        "error":“INVALID_PARAMETER”
}
```

4.4 Auto Sense

4.4.1 Command Usage

auto_sense {on|off} {priorities}

4.4.2 Command Description

This command can be used to set auto sense on or off. Once auto sense is enabled local and remote video source will be switched to the newly plugged source. Or when a source is disconnected the available video source will be routed to video output port if available.

4.4.3 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.4.4 Example

auto_sense on 0 1 2

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

auto_sense off

```
{
        "status":“SUCCESS”,
```

```
        "error":“NULL”
}
```

4.5 Front panel lock

4.5.1 Command Usage

front_panel_lock {on|off}

4.5.2 Command Description

This command locks and unlocks the buttons on the VLX device.

4.5.3 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.5.4 Example

```
front_panel_lock on
{
        "status":“SUCCESS”,
        "error":“NULL”
}

front_panel_lock off
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.6 Update Settings

4.6.1 Command Usage

vlx_config upload <file_name> <settings_type> <ip_address>

4.6.2 Command Description

This command is used to clone the settings of one VLX device to another device or same device later. For this, the settings file should have been saved from the device earlier. Once the settings file is applied, the VLX reboots.

4.6.3 Prerequisite

TFTP server should be running in the PC and the settings file should be placed in the TFTP folder.

4.6.4 Arguments

There are 5 different arguments for settings type in the settings file.

recovery – Applies all the settings from the settings file

encoder – Applies the settings related to encoder only except ip address

decoder – Applies settings corresponding to decoder alone except the ip address

enc/dec – Applies the settings corresponding to Encoder and Decoder except ip address

edid – Applies the EDID settings if the device settings file is downloaded from an encoder and the applied device is in encoder mode.

file_name – File name of the settings

4.6.5 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.6.6 Example

vlx_config upload vlx.config recovery 169.254.2.222

```
{
        "status":“PROCESSING”,
        "error":“NULL”
}
```

vlx_config upload vlx.config encoder 169.254.2.222

```
{
        "status":“PROCESSING”,
        "error":“NULL”
}
```

Invalid file or IP
vlx_config upload vlx1.config encoder 169.254.2.2

```
{
        "status":“ERROR”,
        "error":“INVALID_PARAMETER”
}
```

4.7 Backup Settings

4.7.1 Command Usage

vlx_config download <ip_address>

4.7.2 Command Description

This command is used to download or save the settings of current VLX device so that this settings could be applied to another VLX device or same VLX device later. The downloaded file will be name as vlx.config. The file will be download only in Windows OS and not in linux versions.

4.7.3 Prerequisite

TFTP server should be running in the PC and the settings file will be downloaded to the TFTP folder.

4.7.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.7.5 Example

vlx_config download 169.254.2.222

```
{
        "status":“PROCESSING”,
        "error":“NULL”
}
```

4.8 No Signal Image

4.8.1 Command Usage

- **bitmap upload <file_name> <ip_addr>**
- **bitmap list**
- **bitmap apply <file_name>**
- **bitmap usage**
- **bitmap delete <file_name>**
- **bitmap erase**

4.8.2 Command Description

Bitmap upload – Uploads the bitmap image.

TFTP server should be running in the PC. Image file should be placed in the tftp folder. File in the tftp folder will be uploaded to the VLX device provided the uploaded file is a “.jpg” file extension.

Bitmap list – Lists the images already present in the VLX

Bitmap apply – Applies the selected image as the background image of no signal.

Bitmap delete – Deletes the selected image from the vlx. If the selected image is the current background image it will be removed on next reboot or on applying a different image as background image.

Bitmap erase – Deletes all the images stored in the disk

4.8.3 Return Value

bitmap upload <file_name> <ip_addr>

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

bitmap list

```
{
        "status":"SUCCESS",
        "result": "<files separated by spaces>",
        "error":"NULL"
}
```

bitmap apply <file_name>

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

bitmap usage

```
{
        "status":"SUCCESS",
        "result":"4955,1014,3685,22%",
        "error":"NULL"
}
```

bitmap delete <file_name>

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

bitmap erase

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

4.8.4 Example

bitmap upload "1.jpg" 169.254.2.222

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

bitmap list

```
{
        "status":"SUCCESS",
        "result":"1.jpg",
        "error":"NULL"
}
```

bitmap apply "1.jpg"

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

bitmap usage

```
{
        "status":"SUCCESS",
        "result":"4955,1014,3685,22%",
        "error":"NULL"
}
```

bitmap delete "1.jpg"

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

When any previous command operation is in progress.

bitmap apply "1.jpg"

```
{
        "status":"BUSY",
        "error":"NULL"
}
```

bitmap erase

```
{
        "status":"PROCESSING",
        "error":"NULL"
}
```

4.9 Get

4.9.1 Command usage

get subset

4.9.2 Description

The get command fetches information from the device. For each device, the retrieved information is a subset of the device object, which contains all configurations, status, supported feature and other information about the device.

4.9.3 Subset

The valid values for the ***subset*** and their meaning are given in the table below.

Subset name	Description
Status	Shows connection status of device.
Settings	Return detailed information regarding the current configuration and state of the device (the “device” object itself is returned for this option)
local_display_source	Returns which video source is connected to “HDMI output” of the device.
stream_source	This command is valid only for “Encoders”. Returns which “Video_IN” source is connected to streaming output.
auto_sense	Gets the current status of auto sense. ie;, on or off.
dante	Returns the status of dante audio status ie; on or off.
hdcp_capability	This command is valid only for “Encoders”. Returns the current HDCP version supported by device.
analog_audio_output	This command is valid only for “Encoders”. Returns the status of hdmi audio routing to line out.
bitrate_limit	This command is valid only for “Encoders”. Returns the current status of hdmi audio bitrate limit.
serial_port_wp	Returns the debug port/ soip port of the wallplate. This command is applicable only for wallplate version.

4.9.4 Return Value

```
{
        "status": “string”,
        “result”: “varies”,
        "error": “string”
}
```

The result field varies for different get commands.

4.9.4.1 Status

Result field of “get status” command has any one of the following values, depending on the state of the state machine.

- s_init: This state occurs when the system finished initializations.
- s_idle: In this state system is idle and not searching for connections.

- s_attaching: This is an encoder specific state. It occurs when encoder is connecting to decoder or encoder is already connected to decoder and waiting for video to start.
- s_srv_on: Connection is established between an encoder and decoder and video is available.
- s_search: This is a decoder specific state. It occurs when decoder is connecting to an encoder.
- s_start_srv_lp/s_start_srv_hp: This state occurs when a device is starting to connect.
- s_stop:This state occurs when the connection is stopped.
- s_error: This state occurs if any error happens

4.9.4.2 Settings

Result field of “get settings” command is JSON structured.

Mac	Mac Address in 12-digit hex code
Hostname	Device_id as text string up to 24 characters
FW Version	Version of current firmware
Display_mode	Display mode is either Genlocked or Video wall
Hdmi.tx	y-when device is encoder,n-when device is receiver
Hdmi.hdcp	On- when hdcp is on, off-when hdcp is off
Hdmi.hpd	For future implementation
Hdmi.video_stable	For future implementation
Hdmi.local_display_source	Soucre connected to local display. IN_PORT1-when HDMI_IN1,IN_PORT2-when HDMI_IN2, STREAM-when streaming
Hdmi.stream_source	Source connected to stream out. IN_PORT1-when HDMI_IN1,IN_PORT2-when HDMI_IN2
Ip.mode	IP mode is either ‘DHCP’ ,’static’ or ‘auto’
Ip.address	IPV4 address represented as xxx.xxx.xxx.xxx
Ip.mask	IPV4 subnet mask represented in decimal as xxx.xxx.xxx.xxx
Ip.gateway	IPV4 gateway address represented in decimal as xxx.xxx.xxx.xxx
Streams.video.ip	The IPV4 multicast address used to transmit video. For Encoders it is the address that the device is sending the data to. For Decoders it is the address that the device is registered to receive from.
Streams.video.status	Streaming status. Either ‘streaming’ or ‘stopped’
Streams.video.wallsize_r	Wall size(no of rows)
Streams.video.wallsize_c	Wall size (no of columns)
Streams.video.row_idx	Row position of the device
Streams.video.col_idx	Column position of the device

Streams.video.overall_width	Overall width of display
Streams.video.view_width	View width of display
Streams.video.overall_height	Overall height of display
Streams.video.view_height	View height of display
Streams.video.h_scale	Horizontal scale in number of pixels
Streams.video.v_scale	Vertical scale in number of pixels
Streams.video.h_shift	Horizontal shift
Streams.video.v_shift	Vertical shift
Streams.video.rotate	Rotation applied to image. Either 0, 180 or 270
Streams.video.frame_rate	Frame rate of the encoder
Streams.video.bit_rate	Encoder profile in Mbps
Streams.video.h_size	Horizontal video resolution on the HDMI input (for encoder) or HDMI output (for decoder) connector
Streams.video.v_size	Vertical video resolution on the HDMI input (for encoder) or HDMI output (for decoder) connector
Streams.video.fps	Video frame rate on the HDMI input (for encoder) or HDMI output (for decoder) connector
Streams.rs232.soip_status	On- if soip is enabled, off- if soip disabled
Streams.rs232.baudrate	Rs-232 Baud rate. Max is 115200
Streams.rs232.data_bit	Rs-232 data bit. Either 5,6, 7 or 8 bits
Streams.rs232.stop_bit	Rs-232 stop bit. Either set to 1 or 0
Streams.rs232.parity	Rs-232 parity bit. Either ‘none’, ‘even’ or ‘odd’
Streams.rs232.ip	For future implementation
Streams.rs232.mode	Mode in which soip is operating (telnet or redirection)

4.9.4.3 Local_display_source

The result field for “get local_display_source” takes any one of the following values.

- IN_PORT1-when HMDI_IN1 is connected to HDMI_OUT
- IN_PORT2-when HDMI_IN2 is connected to HDMI_OUT
- STREAM- when streaming. This value is possible only if the device is a decoder.

4.9.4.4 Stream_source

The result field for “get stream_source” takes any one of the following values

- IN_PORT1-when HDMI_IN1 is stream source
- IN_PORT2-when HDMI_IN2 is stream source

This subset is relevant for an encoder only

4.9.5 Example

See examples in Appendix A.

4.10 Join

4.10.1 Command usage

join { “HDMI” } {host name} [“display”]

4.10.2 Description

This command can be used to subscribe one or more VLX decoder units to a specific VLX encoder. Before joining HDMI video (i.e. stream type HDMI), the display mode and video output parameters must have been set on the decoder device(s) as compatible to that of the stream. This command applies only to decoder device.

4.10.3 Arguments

- First argument must be any one of the stream type.
  HDMI for an HDMI video stream. (It is the only stream type currently supported)
- The **host_name** specifies the device_id of the streaming encoder.
- **display** is an optional argument. If this argument is specified the decoder will automatically select the local display source as STREAM. If this is not specified, “set local_display_soucre STREAM” command has to be given after giving “join” command

4.10.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.10.5 Example

join HDMI vlx-series-enc-1234

```
{
  "status":“SUCCESS”,
  "error":“NULL”
}
```

join HDMI vlx-series-enc-1234 display

```
{
      "status":“SUCCESS”,
      "error":“NULL”
}
```

4.11 Leave

4.11.1 Command usage

leave {“HDMI” }

4.11.2 Description
Unsubscribe the device from a transmitter device stream. This command applies only to decoder device

4.11.3 Arguments
The argument must be a subscription type. The following types are supported:

- HDMI for an HDMI video stream

Note: Currently only Video stream is supported. Other stream types such as AUDIO, USB, IR etc. will be added in later versions.

4.11.4 Return Value
```
{
        "status":“string”,
        "error":“string”
}
```

4.11.5 Example
leave HDMI
```
{
         "status":"SUCCESS",
         "error":“NULL”
}
```

4.12 Reboot

4.12.1 Command usage
reboot

4.12.2 Description
Causes the device to restart. The device becomes unavailable for a short period while it restarts.

4.12.3 Return Value
```
{
        "status":“string”,
        "error":“string”
}
```

4.12.4 Example
reboot
```
{
       "status":"SUCCESS",
       "error":“NULL”
}
```

4.13 Update Firmware

4.13.1 Command usage

update_fw {filename} {ip_address}

4.13.2 Description

This commands updates the firmware of the device using the update file present in the tftpd current directory of the machine whose ip address is provided. Once update has started no other commands will be accepted by the device and will return error status.

4.13.3 Arguments

filename: Name of the file in the machine whose ip address is mentioned in the command.

ip_address: IP address of the machine from which TFTP has to be performed

4.13.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.13.5 Example

update_fw VLX-FW-CR_2_2_2.bin 169.254.1.10

```
{
      "status":"PROCESSING",
      "error":“NULL”
}
```

on inputting any other command while firmware update is in process

```
{
        "status":“ERROR”,
        "error":“FW_UPDATE_IN_PROCESS”
}
```

4.14 Send

The send command provides sub-commands which can be used to send data to destination device.

4.14.1 send IR

4.14.1.1 Command usage

send IR *{data_hex}*

4.14.1.2 Description

Send infrared remote control data to the device.

4.14.1.3 Arguments

- **data_hex** is the hex value to be passed to destination.

Note: This command is not implemented in the current version. Minor changes may be expected when implementing.

4.14.2 send edid

4.14.2.1 Command usage

send edid

4.14.2.2 Description

Send EDID of decoder to the remote encoder. This command is applicable only for decoders.

4.14.2.3 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.14.2.4 Example

```
send edid
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.14.3 send RS232

4.14.3.1 Command usage

send RS232 ‘string’ {baudrate} {databits}

4.14.3.2 Description

Sends the string in single quotes with the supplied baud-rate and data-bit, to the rs232 port of the VLX device. All the escape character checks are performed. This command returns success only if the device has the SOIP disabled. The string should be always enclosed in a single quote, otherwise no escape sequences will be parsed.

The parser shall check for the following sequence of characters in the receive packet and replace with respective hex code and send to local RS232 out port
\0 - null character (0x00)
\n - line feed (0x0a)
\r - carriage return (0x0d)
\t - horizontal tab (0x09)
\xnn - (where nn are two hexadecimal characters) hexadecimal representation of byte.
There should be 2 characters after \x
\\ - a single backslash
\(space) - space character(0x20)

4.14.3.3 Arguments

String: Should be given in single quotes

baudrate: Supported baudrates are 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

databits: Supported data bits are 7 and 8

4.14.3.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.14.3.5 Example

```
send RS232 'my_string' 115200 8
{
        "status":"SUCCESS",
        "error":“NULL”
}
```

4.15 Set

4.15.1 set ip

4.15.1.1 Command usage

set ip {“auto”|”static”|”DHCP”} [ipaddr netmask gatewayip]

4.15.1.2 Description

Set the Internet Protocol (IP) configuration of the device. The configuration change will come to effect only on reboot.

4.15.1.3 Arguments

- static for manual IP configuration
- DHCP for automatic configuration using a DHCP server.

- auto for automatic ip generation.

If mode is static, the ipaddr, netmask and gatewayip arguments must be specified. If mode is not static, these arguments must not be specified.

4.15.1.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.1.5 Example

set ip auto

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

set ip static 165.254.10.110 255.255.0.0 169.254.0.254

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

set ip DHCP

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

Note: After changing ip address, the device will only respond to its new IP address. So the application SW would need to “re-discover” the device to get access to using its new IP address.

4.15.2 set local_display_source

4.15.2.1 Command usage

set local_display_source {IN_PORT1 | IN_PORT2 |STREAM}

4.15.2.2 Description

Select the source for HDMI output of the device.

4.15.2.3 Arguments

- IN_PORT1 selects Video IN_port_1 (HDMI1) as source for HDMI out.
- IN_PORT2 selects Video IN_ port_2 (HDMI2) as source for HDMI out.
- STREAM selects streaming input as source for HDMI out. This argument can be used only for decoders.

4.15.2.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.2.5 Example

set local_display_source STREAM
Result for decoder:

```
{
        "status":"SUCCESS",
        "error":“NULL”
}
```

Result for encoder :

```
{
        "status":"ERROR”,
        "error":“INVALID_PARAMETER”
}
```

set local_display_source IN_PORT1
Result for decoder and encoder:

```
{
        "status":"SUCCESS",
        "error":“NULL”
}
```

4.15.3 set stream_source

4.15.3.1 Command usage

set stream_source {IN_PORT1 | IN_PORT2 }

4.15.3.2 Description

Select the source for streaming output for an encoder. This command is valid only for Encoder devices.

4.15.3.3 Arguments

- IN_PORT1 selects Video IN port 1 (HDMI-1) as source for Video stream out.
- IN_PORT2 selects Video IN port 2 (HDMI-2) as source for Video stream out.

4.15.3.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.3.5 Example

set stream_source IN_PORT1

```
{
        "status":"SUCCESS",
        "error":“NULL”
}
```

4.15.4 set videooip

4.15.4.1 Command usage

set videooip {profile: *value*} {frame_rate: *value*}

4.15.4.2 Description

Set video over ip parameters for an encoder.

4.15.4.3 Arguments

- Profile: Encoder profile (in Mbps). It can take the following values
  - 0,10,50,100,150,200

  If profile is set to the value “0” then bitrate will be selected automatically.
- frame_rate-varies from 1 to 59

4.15.4.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.4.5 Example

set videooip profile: 200 frame_rate: 50

```
{
        "status":“SUCCESS”
        "error":“NULL”
}
```

4.15.5 set vwall

4.15.5.1 Command usage

set vwall {“ENABLE” | “DISABLE”} {wall_size: *max_row max_col*} {position_index: *row_id col_id*} {bezel: *view_width overall_width view_height overall_height*}

4.15.5.2 Description

Set the display mode and video output parameters of a decoder in video wall configuration. This is applicable only for decoders.

4.15.5.3 Arguments

First argument must be one of the following strings.

- ENABLE
- DISABLE

If first argument is DISABLE, then the rest of the arguments must not be specified.

wall_size specifies video wall size.

- max_row is total no of rows. It can take values from 1 to 8.
- max_col is total no of columns. It can take values from 1 to 16

position_index specifies the position of decoder in video wall configuration

- row_id is the index of row in which decoder is located. This value can vary from 1 to max_row.
- col_id is the index of column in which decoder is located. This value can vary from 1 to max_col.

bezel specifies bezel and gap compensation. View height should be smaller than over all height and view width should be smaller than over all width. If bezel and gap compensation is not to be specified all the four arguments should be set to 1.

4.15.5.4 Return Value

```
{
        "status":”string”,
        "error":“string”
}
```

4.15.5.5 Example

set vwall ENABLE wall_size: 2 2 position_index: 1 1 bezel: 100 200 90 99

```
{
          "status":“SUCCESS”,
          "error":“NULL”
}
```

set vwall ENABLE wall_size: 2 2 position_index: 1 1 bezel: 1 1 1 1

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

set vwall DISABLE

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.6 set vw_screen_layout

4.15.6.1 Command usage

set vw_screen_layout {*max_row max_col row_id col_id*}

4.15.6.2 Description

This command can be used only after setting up a video wall configuration using set vwall. This command tells a device to change its screen layout (for example from 3x3 to 2x2). This is applicable only for decoders.

4.15.6.3 Arguments

- max_row is total no of rows
- max_col is total no of columns
- row_id is the new row index of the device
- col_id is the new col index of the device

4.15.6.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.6.5 Example

set vw_screen_layout 1 1 1 1

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.7 set vw_scale

4.15.7.1 Command usage

set vw_scale {*hscale vscale*}

4.15.7.2 Description

This command is used on an existing video wall configuration to fine tune vertical or horizontal scaling. This is applicable only for decoders.

4.15.7.3 Arguments

- hscale –horizontally scale up “hscale” number of pixels.
- vscale - vertically scale up “vscale” number of pixels.

4.15.7.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.7.5 Example

set vw_scale 4 0

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

set vw_scale 1 2

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.8 set vw_hshift

4.15.8.1 Command usage

set vw_hshift { “left” | “right” } {value}

4.15.8.2 Description

This command is used on an existing video wall configuration to shift the image in a desired direction. 8 pixels align. This is applicable only for decoders.

4.15.8.3 Arguments

First argument selects whether shift is to be in left or right direction and ***value*** specifies the number of pixels to be shifted.

4.15.8.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.8.5 Example

set vw_hshift left 200

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.9 set vw_vshift

4.15.9.1 Command usage

set vw_vshift { “up” | “down” } {value}

4.15.9.2 Description

This command is used on an existing video wall configuration to shift the image in a desired direction. 8 pixels align. This is applicable only for decoders.

4.15.9.3 Arguments

First argument selects whether shift is to be in up or down direction and ***value*** specifies the number of pixels to be shifted.

4.15.9.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.8.9.5 Example

set vw_vshift up 200

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.10 set vw_rotate

4.15.10.1 Command usage

set vw_rotate { 0 | 180 | 270 }

4.15.10.2 Description

This command is used on an existing video wall configuration to rotate an image.

4.15.10.3 Arguments

This command takes either one of the following arguments.

- 0 –no rotation
- 180 -rotate 180^0
- 270 - rotate 270^0

4.15.10.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.10.5 Example

set vw_rotate 180

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.11 set hostname_prefix

4.15.11.1 Command usage

set hostname_prefix {“prefix string”}

4.15.11.2 Description

This command set “device name” of VLX device to the user specified string.

4.15.11.3 Arguments

A string of maximum 20 alphanumeric characters including “-“. -dec-‘ or ‘-enc-‘ part of the device name changes automatically depending on whether the device is configured as encoder or decoder. This part cannot be changed using “set hostname_prefix” command.

4.15.11.4 Return Value

```
{
        "status":“string”,
        “error":“string”
}
```

4.15.11.5 Example

set hostname_prefix vlx-series

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.12 set hostname_id

4.15.12.1 Command usage

set hostname_id {host_id}

4.15.12.2 Description

This command sets the hostname prefix of the VLX device to the set value.

4.15.12.3 Arguments

host_id is a four-digit number starting from 0000 to 9999. It doesn't accept any other characters and the minimum number of digits must be four.

4.15.12.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.12.5 Example

set hostname_id 1234

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.13 set led

4.15.13.1 Command usage

set led {“blink” | “normal”}

4.15.13.2 Description

This command will make ir_tx and ir_rx LEDs to blink or turn off so that the device can be identified in a network.

4.15.13.3 Arguments

- **blink** makes both ir leds to blink continuously.
- **normal** turns off the blinking ir leds.

4.15.13.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.13.5 Example

set led blink

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

set led normal
{
"status":“SUCCESS”,
"error":“NULL”
}

4.15.14 set mode

4.15.14.1 Command usage

set mode {“encoder” | “decoder”}

4.15.14.2 Description

This command makes the device to switch to encoder mode if it is a decoder or to decoder it is in encoder mode.

4.15.14.3 Arguments

- **encoder** sets the device to encoder. This automatically reboots the board on receiving the command if it is a decoder. If it is already encoder it does nothing but returns a success status
- **decoder** sets the device to decoder. This automatically reboots the board on receiving the command if it is an encoder. If it is already decoder, it does nothing but returns a success status

4.15.14.4 Return Value

{
"status":“string”,
"error":“string”
}

4.15.14.5 Example

set mode encoder
{
"status":“SUCCESS”,
"error":“NULL”
}

set mode decoder
{
"status":“SUCCESS”,
"error":“NULL”
}

set mode decode
{
“status”:”ERROR”,
“error”:”INVALID_PARAMETER”
}

4.15.15 set audio_input

4.15.15.1 Command usage

set audio_input {“HDMI” | “line_in” | “none”}

4.15.15.2 Description

This command is used to set the audio source. This command is applicable only for encoder.
Note: This setting will only work if input HDMI contains only **48khz** audio.

4.15.15.3 Arguments

- **HDMI** sets the audio source as HDMI audio. The HDMI audio will be routed to remote device
- **line_in** sets the audio source as line in. Line in audio is routed to remote device.
- **none** is specified when no audio source is required. No audio is routed to remote device.

4.15.15.4 Return Value

{
"status":“string”,
"error":“string”
}

4.15.15.5 Example

set audio_input HDMI
{
"status":“SUCCESS”,
"error":“NULL”
}

4.15.16 set bitrate_limit

4.15.16.1 Command usage

set bitrate_limit {on | off}

4.15.16.2 Description

This command limits the audio sampling rate to 96 kHz. This command is applicable only for encoders.

4.15.16.3 Arguments

- **on** limits audio sampling rate to 96 kHz.
- **off** disables the sampling rate limit.

4.15.16.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.16.5 Example

set bitrate_limit on

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.17 set dante

4.15.17.1 command usage

set dante {HDMI | line_in | none}

4.15.17.2 Description

This command is used to set the dante input. This command is applicable for encoder and decoder.

4.15.17.3 Arguments

- HDMI - HDMI input audio is routed through dante. Audio from remote is always routed to lineout
- line_in – Line in is routed through dante. Audio from remote is always routed to lineout.
- none - No audio input is connected to dante. Audio from remote is always routed to lineout.

4.15.17.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.17.5 Example

set dante HDMI

```
{
"status":“SUCCESS”,
"error":“NULL”
}
```

set dante line_in
{
"status":“SUCCESS”,
"error":“NULL”
}

set dante none
{
"status":“SUCCESS”,
"error":“NULL”
}

4.15.18 set analog_audio_output

4.15.18.1 Command Usage

set analog_audio_output {on|off}

4.15.18.2 Description

This command is used the set HDMI to lineout in encoder. This command is valid only in encoder.

4.15.18.3 Arguments

- **on** Enables HDMI to lineout
- **off** Disables HDMI to lineout

4.15.18.4 Return value

{
"status":"SUCCESS",
"error":"NULL"
}

4.15.18.5 Example

set analog_audio_output off
{
"status":"SUCCESS",
"error":"NULL"
}
set analog_audio_outpu off
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}

4.15.19 set linein_lineout

4.15.19.1 Command Usage

set linein_lineout {on|off}

4.15.19.2 Description

This command is used the set LINEIN to lineout in encoder. This command is valid only in encoder.

4.15.19.3 Arguments

- **on** Enables Line In to lineout
- **off** Disables Line In to lineout

4.15.19.4 Return value

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

4.15.19.5 Example

```
set linein_lineout off
{
"status":"SUCCESS",
"error":"NULL"
}
set linein_lineou off
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}
```

4.15.20 set dante_lineout

4.15.20.1 Command Usage

set dante_lineout {on|off}

4.15.20.2 Description

This command is used the set Dante to lineout in encoder. This command is valid only in encoder.

4.15.20.3 Arguments

- **on** Enables Dante to lineout
- **off** Disables Dante In to lineout

4.15.20.4 Return value

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

4.15.20.5 Example

set dante_lineout off

```
{
"status":"SUCCESS",
"error":"NULL"
}
set dante_lineou off
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}
```

4.15.21 set hdmi_out_audio

4.15.21.1 Command Usage

set hdmi_out_audio {HDMI|none|Dante}

4.15.21.2 Description

This command is used the set HDMI audio output off decoder. This command is valid only in decoder.

4.15.21.3 Arguments

- **HDMI** Enables HDMI to HDMI of encoder
- **none** Disables HDMI of decoder
- **Dante** Enables Dante to HDMI of encoder

4.15.21.4 Return value

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

4.15.21.5 Example

```
set hdmi_out_audio off
{
"status":"SUCCESS",
"error":"NULL"
}
set hdmi_out_audi off
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}
```

4.15.22 set stream_lineout

4.15.22.1 Command Usage

set stream_lineout {on|off}

4.15.22.2 Description

This command is used the set HDMI to lineout in Decoder. This command is valid only in decoder.

4.15.22.3 Arguments

- **on** Enables HDMI to lineout
- **off** Disables HDMI to lineout

4.15.22.4 Return value

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

4.15.22.5 Example

```
set stream_lineout off
{
"status":"SUCCESS",
"error":"NULL"
}
set stream_lineou off
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}
```

4.15.23 set linein_lineout

4.15.23.1 Command Usage

set linein_lineout {on|off}

4.15.23.2 Description

This command is used the set Line in to lineout in Decoder. This command is valid only in decoder.

4.15.23.3 Arguments

- **on** Enables Line in to lineout
- **off** Disables Line in to lineout

4.15.23.4 Return value

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

4.15.23.5 Example

set linein_lineout off

{
"status":"SUCCESS",
"error":"NULL"
}
set linein_lineou off
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}

4.15.24 set dante_lineout

4.15.24.1 Command Usage

set dante_lineout {on|off}

4.15.24.2 Description

This command is used the set dante to lineout in Decoder. This command is valid only in decoder.

4.15.24.3 Arguments

- **on** Enables dante to lineout
- **off** Disables dante to lineout

4.15.24.4 Return value

{
"status":"SUCCESS",
"error":"NULL"
}

4.15.24.5 Example

set dante_lineout off
{
"status":"SUCCESS",
"error":"NULL"
}
set dante_lineou off
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}

4.15.25 set linein_volume

4.15.25.1 Command usage

set linein_volume {0 to 10}

4.15.25.2 Description

This command is used to set the audio Line in volume level. This command is applicable only for encoders.
Note: This setting will only work if input HDMI contains only **48khz** audio.

4.15.25.3 Arguments

- The volume level is specified in range **0 to 10**.
- Or **"mute"** : This will mute the line in signal.
- **"unmute"** to unmute the line in signal.

4.15.25.4 Return Value

```
{
        "status":"string",
        "error":"string"
}
```

4.15.25.5 Example

set linein_volume 4

```
{
        "status":"SUCCESS",
        "error":"NULL"
}
```

4.15.26 set lineout_volume

4.15.26.1 Command usage

set lineout_volume {0 to 10}

4.15.26.2 Description

This command is used to set the audio line out volume level. This command is applicable only for decoders.

4.15.26.3 Arguments

- The volume level is specified in range **0 to 10**.
- Or **"mute"** : This will mute the line out signal.
- **"unmute"** to unmute the line in signal.

4.15.26.4 Return Value

```
{
        "status":"string",
        "error":"string"
}
```

4.15.26.5 Example

set lineout_volume 7

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.27 set hdcp_capability

4.15.27.1 Command usage

set hdcp_capabililty {all | hdcp14 | none}

4.15.27.2 Description

Select the HDCP version of the device for Encoder mode. This command is valid only for Encoder devices.

4.15.27.3 Arguments

- **all** - Enables HDCP 2.2 Type 0 and HDCP 1.4.
- **hdcp14** - Enables HDCP 1.4 and disables HDCP 2.2.
- **none** - Disables HDCP capability of the device.

4.15.27.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.27.5 Example

set hdcp_capability hdcp14

```
{
        "status":"SUCCESS",
        "error":“NULL”
}
```

4.15.28 set analog_audio_output

4.15.28.1 Command usage

set analog_audio_output {on | off}

4.15.28.2 Description

This command is used to enable/disable routing of HDMI input audio into Line Out. This command is applicable only for encoders.

4.15.28.3 Arguments

on - enables routing of HDMI audio into Line Out.
off - disables routing of HDMI audio into Line Out.

4.15.28.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.28.5 Example

set analog_audio_output on

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

set analog_audio_output off

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.29 set mode toggle

4.15.29.1 Command Usage

set mode toggle

4.15.29.2 Description

Toggles encoder decoder mode.

4.15.29.3 Return Value

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.29.4 Example

set mode toggle

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.15.30 serial_port_wp

4.15.30.1 Command Usage

set serial_port { front | rear }

4.15.30.2 Description

Sets the serial console/soip port of VLX wallplate to front port or rear port. This command is applicable only for wallplate. This command is not applicable when SoIP is enabled.

4.15.30.3 Arguments

front – sets the serial debug console/soip port to front

rear – sets the serial debug console / soip port to rear.

4.15.30.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.30.5 Examples

set serial_port_wp front

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

set serial_port_wp rear

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

In Box version

set serial_port_wp front

```
{
        "status":“ERROR”,
        "error":“INVALID_PARAMETER”
}
```

4.15.31 ir_auto_pair

4.15.31.1 Command Usage

set ir_auto_pair { enable| disable }

4.15.31.2 Description

Sets the value of IR Auto Pair. This command is applicable only for decoder.

4.15.31.3 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.31.4 Examples

set ir_auto_pair enable

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

set ir_auto_pair disable

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

set ir_auto_pair off

```
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}
```

4.15.32 vlan enable

4.15.32.1 Command Usage

set vlan enable <vp vid> <dante vid>

4.15.32.2 Description

Enables vlan in the device. Also we can able to set video port vlan id and dante port vlan id. This command is applicable both in encoder and decoder.

4.15.32.3 Return Value

```
{
        "status":“string”,
        "error":“string”
```

```
}
```

4.15.32.4 Examples

```
set vlan enable 5 10
{
"status":"SUCCESS",
"error":"NULL"
}
```

4.15.33 vlan disable

4.15.33.1 Command Usage

set vlan disable

4.15.33.2 Description

Disables vlan in the device. This command is applicable both in encoder and decoder.

4.15.33.3 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.15.33.4 Examples

```
set vlan disable
{
"status":"SUCCESS",
"error":"NULL"
}
```

4.16 chmap_upload

4.16.1 Command Usage

chmap upload <file_name> <ip_addrs>

4.16.2 Description

Uploads chmap file into the board. This command is valid only in decoder.

4.16.3 Prerequisite

TFTP server should be running in the PC and the chmap csv file should be placed in the TFTP folder.

4.16.4 Arguments

file_name – File name of the settings

ip_addrs – IP Address of the system where TFTP server is running.

4.16.5 Return value and Example

chmap upload map_0023.csv 169.254.2.222

```
{
        "status":“PROCESSING”,
        "error":“NULL”
}
```

Invalid file or IP

chmap upload map 0023.csv 169.254.2.222

```
{
        "status":“ERROR”,
        "error":“INVALID_PARAMETER”
}
```

4.17 chmap <disable/enable>

4.17.1 Command Usage

chmap <disable/enable>

4.17.2 Description

We can enable or disable the chmap feature. This command is valid only in decoder.

4.17.3 Arguments

enable – enable chmap feature.

disable – disables chmap feature

4.17.4 Return value and Example

chmap enable

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

chmap disable

```
{
"status":"SUCCESS",
"error":"NULL"
}
```

chamap disble

```
{
"status":"ERROR",
"error":"INVALID_COMMAND"
}
```

4.18 chmap_ir <value>

4.18.1 Command Usage

chmap_ir <value>

4.18.2 Description

We can set the chmap_ir value by using the commands. This command is valid only in decoder.

4.18.3 Arguments

<value> -

0 Disable,

1 - Aurora remote,

2 Philips remote

4.18.4 Return value and Example

chmap_ir 0
{
"status":"SUCCESS",
"error":"NULL"
}
chmap_ir 1
{
"status":"SUCCESS",
"error":"NULL"
}
chmap_ir 2
{
"status":"SUCCESS",
"error":"NULL"
}
champ_ir 3
{
"status":"ERROR",
"error":"INVALID_COMMAND"
}

4.19 preview start/stop

4.19.1 Command Usage

preview <start/stop>

4.19.2 Description

We can start or stop the preview feature. This command is valid in encoder and decoder.

4.19.3 Arguments

start – start preview feature.

stop – disables preview feature

4.19.4 Return value and Example

preview start
{
"status":"PROCESSING",
"error":"NULL"
}
preview stop
{
"status":"PROCESSING",
"error":"NULL"
}
prview stop
{
"status":"ERROR",
"error":"INVALID_COMMAND"
}

4.20 preview mode <quality> <fps>

4.20.1 Command usage

preview mode <quality> <fps>

4.20.2 Arguments

Quality – We can select quality as Low or High

Fps – We can select fps from 1 to 10.

4.20.3 Return value and Example

preview mode High 10
{
"status":"SUCCESS","error":"NULL"
}
preview mode low 7
{
"status":"SUCCESS","error":"NULL"
}

4.21 send edid <value>

4.21.1 Command usage

send edid <value>

4.21.2 Description

We can send the edid of the monitor connected in the decoder to encoder. This command is valid only in decoder

4.21.3 Arguments

<value> - "0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x17, 0x10, 0x01, 0x09, 0x01, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x01, 0x03, 0x80, 0x73, 0x41, 0x78, 0x2a, 0x7c, 0x11, 0x9e, 0x59, 0x47, 0x9b, 0x27, 0x10, 0x50, 0x54, 0x21, 0x08, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x8c, 0x0a, 0xd0, 0x8a, 0x20, 0xe0, 0x2d, 0x10, 0x10, 0x3e, 0x96, 0x00, 0x04, 0x03, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x45, 0x50, 0x2d, 0x48, 0x44, 0x4d, 0x49, 0x2d, 0x52, 0x58, 0x0a, 0x20, 0x20, 0x00, 0x00, 0x00, 0xfd, 0x00, 0x3b, 0x3d, 0x0f, 0x2e, 0x08, 0x00, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x01, 0xab, 0x02, 0x03, 0x34, 0x71, 0x4a, 0x84, 0x13, 0x05, 0x14, 0x10, 0x02, 0x03, 0x11, 0x12, 0x01, 0x38, 0x09, 0x0f, 0x07, 0x0f, 0x7f, 0x07, 0x15, 0x07, 0x50, 0x3e, 0x06, 0xc0, 0x4d, 0x02, 0x00, 0x57, 0x06, 0x01, 0x5f, 0x7f, 0x01, 0x67, 0x7f, 0x01, 0x83, 0x4f, 0x00, 0x00, 0x67, 0x03, 0x0c, 0x00, 0xff, 0xff, 0x80, 0x1e, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x00, 0xbc, 0x52, 0xd0, 0x1e, 0x20, 0xb8, 0x28, 0x55, 0x40, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x80, 0x18, 0x71, 0x1c, 0x16, 0x20, 0x58, 0x2c, 0x25, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x01, 0x1d, 0x80, 0xd0, 0x72, 0x1c, 0x16, 0x20, 0x10, 0x2c, 0x25, 0x80, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x00, 0x00, 0x00, 0xd6,"

4.21.4 Return value and Example

send edid "0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x17, 0x10, 0x01, 0x09, 0x01, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x01, 0x03, 0x80, 0x73, 0x41, 0x78, 0x2a, 0x7c, 0x11, 0x9e, 0x59, 0x47, 0x9b, 0x27, 0x10, 0x50, 0x54, 0x21, 0x08, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x8c, 0x0a, 0xd0, 0x8a, 0x20, 0xe0, 0x2d, 0x10, 0x10, 0x3e, 0x96, 0x00, 0x04, 0x03, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x45, 0x50, 0x2d, 0x48, 0x44, 0x4d, 0x49, 0x2d, 0x52, 0x58, 0x0a, 0x20, 0x20, 0x00, 0x00, 0x00, 0xfd, 0x00, 0x3b, 0x3d, 0x0f, 0x2e, 0x08, 0x00, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x01, 0xab, 0x02, 0x03, 0x34, 0x71, 0x4a, 0x84, 0x13, 0x05, 0x14, 0x10, 0x02, 0x03, 0x11, 0x12, 0x01, 0x38, 0x09, 0x0f, 0x07, 0x0f, 0x7f, 0x07, 0x15, 0x07, 0x50, 0x3e, 0x06, 0xc0, 0x4d, 0x02, 0x00, 0x57, 0x06, 0x01, 0x5f, 0x7f, 0x01, 0x67, 0x7f, 0x01, 0x83, 0x4f, 0x00, 0x00, 0x67, 0x03, 0x0c, 0x00, 0xff, 0xff, 0x80, 0x1e, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x00, 0xbc, 0x52, 0xd0, 0x1e, 0x20, 0xb8, 0x28, 0x55, 0x40, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x80, 0x18, 0x71, 0x1c, 0x16, 0x20, 0x58, 0x2c, 0x25, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x01, 0x1d, 0x80, 0xd0, 0x72, 0x1c, 0x16, 0x20, 0x10, 0x2c, 0x25, 0x80, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x00, 0x00, 0x00, 0xd6,"

{
"status":"SUCCESS",
"error":"NULL"
}

send edod "0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x17, 0x10, 0x01, 0x09, 0x01, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x01, 0x03, 0x80, 0x73, 0x41, 0x78, 0x2a, 0x7c, 0x11, 0x9e, 0x59, 0x47, 0x9b, 0x27, 0x10, 0x50, 0x54, 0x21, 0x08, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x8c, 0x0a, 0xd0, 0x8a, 0x20, 0xe0, 0x2d, 0x10, 0x10, 0x3e, 0x96, 0x00, 0x04, 0x03, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x45, 0x50, 0x2d, 0x48, 0x44, 0x4d, 0x49, 0x2d, 0x52, 0x58, 0x0a, 0x20, 0x20, 0x00, 0x00, 0x00, 0xfd, 0x00, 0x3b, 0x3d, 0x0f, 0x2e, 0x08, 0x00, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x01, 0xab, 0x02, 0x03, 0x34, 0x71, 0x4a, 0x84, 0x13, 0x05, 0x14, 0x10, 0x02, 0x03, 0x11, 0x12, 0x01, 0x38, 0x09, 0x0f, 0x07, 0x0f, 0x7f, 0x07, 0x15, 0x07, 0x50, 0x3e, 0x06, 0xc0, 0x4d, 0x02, 0x00, 0x57, 0x06, 0x01, 0x5f, 0x7f, 0x01, 0x67, 0x7f, 0x01, 0x83, 0x4f, 0x00, 0x00, 0x67, 0x03, 0x0c, 0x00, 0xff, 0xff, 0x80, 0x1e, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x00, 0xbc, 0x52, 0xd0, 0x1e, 0x20, 0xb8, 0x28, 0x55, 0x40, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x80, 0x18, 0x71, 0x1c, 0x16, 0x20, 0x58, 0x2c, 0x25, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x01, 0x1d, 0x80, 0xd0, 0x72, 0x1c, 0x16, 0x20, 0x10, 0x2c, 0x25, 0x80, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x00, 0x00, 0x00, 0xd6,"

```
{
"status":"ERROR",
"error":"INVALID_PARAMETER"
}
```

4.22 Start

4.22.1 Command usage

start {“HDMI”}

4.22.2 Description

Starts specified stream on an encoder device.

4.22.3 Arguments

The argument must be a stream type. The following types are supported:

- HDMI for an HDMI video stream

4.22.4 Return value and Example

```
{
        "status":“string”,
        "error":“string”
}
```

4.22.5 Example

start HDMI

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.23 Stop

4.23.1 Command usage

stop {“HDMI”}

4.23.2 Description

Stops specified stream on an encoder device.

4.23.3 Arguments

The argument must be a stream type. The following types are supported:

- HDMI for an HDMI video stream

4.23.4 Return Value

```
{
        "status":“string”,
        "error":“string”
}
```

4.23.5 Example

stop HDMI

```
{
        "status":“SUCCESS”,
        "error":“NULL”
}
```

4.24 Web Authentication

4.24.1 Command usage

- **web_auth enable**
- **web_auth disable**
- **web_auth set <user> <password>**

4.24.2 Description

This command is used for enable and disable the web authentication and to set username and password.

4.24.3 Return value and Example

web_auth enable

{

"status":"SUCCESS","error":"NULL"

}

web_auth disable

{

"status":"SUCCESS","error":"NULL"

}

web_auth set admin password

{

"status":"SUCCESS","error":"NULL"

}

4.25 Genlock enable/disable

4.25.1 Command usage

- **video_genlock enable**
- **video_genlock disable**

4.25.2 Description

This command is used for enable and disable Genlock mode in decoders. This command is only applicable in decoders.

4.25.3 Return value and Example

video_genlock enable

{

"status":"SUCCESS","error":"NULL"

}

video_genlock disable

{

"status":"SUCCESS","error":"NULL"

}

4.26 HDMI CEC Support

4.26.1 Command Usage

- **cec_send tv_on**
- **cec_send tv_src_switch**
- **cec_send tv_off**

4.26.2 Description

This command is used for sending CEC commands. This command is only applicable in decoders.

4.26.3 Return value and Example

cec_send tv_on

{

"status":"SUCCESS","error":"NULL"

}

cec_send tv_src_switch

{

"status":"SUCCESS","error":"NULL"

}

cec_send tv_off

{

"status":"SUCCESS","error":"NULL"

}

4.27 Mute stream output of VLX decoder

4.27.1 Command Usage

- **stream_out on**
- **stream_out off**

4.27.2 Description

This command is used to mute stream output of VLX decoder. This command is only applicable in decoders.

4.27.3 Return value and Example

stream_out on

{

"status":"SUCCESS","error":"NULL"

}

stream_out off

{

"status":"SUCCESS","error":"NULL"

}

stream_out oof

{

"status":"ERROR","error":"INVALID_PARAMETER"

}

4.28 Switch

4.28.1 Command usage

switch {soip: “ENABLE” | “DISABLE”} {baudrate: “value”} {mode: “redirection_server” | ”redirection_client” | ”telnet”} [“host_device_id”]

4.28.2 Description

This command establishes a bidirectional connection between two devices in redirection mode and in the telnet mode any machine in the network can communicate with the vlx device through a telnet connection to the tcp port 6752.

When redirection mode is to be activated, the user should input ***redirection_server*** as mode in one VLX device with host_device_id as the device_id of the the 2nd vlx device to be connected. In the 2nd device the **redirection_client** should be choosen as the mode along with the device id of the 1st device as the host_device_id. Any device can be made server and client. Both baudrates and data bits should be same in the 2 commands

When telnet mode is selected, the user should use separate telnet connection to the devices TCP Port 6752. The telnet session will become the RS232 redirection session immediately using baud rate settings.

4.28.3 Arguments

- **soip** argument takes one of the following values
  - ENABLE
  - DISABLE

If soip argument is DISABLE, then the rest of the arguments must not be specified.

- **baudrate** argument sets the baudrate. For example: "115200-8n1" means using "115200" baudrate with data bits "8", parity "None" and stop bits "1". The maximum supported baudrate is 115200.
  - Baudrate can be: 300,600,1200,2400,4800,9600,19200,38400,57600,115200
  - Data bits can be: 5 or 6 or 7 or 8
  - Parity can be:
    n : None
    e : even
    o : odd
  - Stop bits can be: 1 or 2
- **Mode** argument takes one of the following values
  - **redirection_server**
  - **redirection_client**
  - **telnet**

  One device should act as redirection_sever and the other device in redirection_client for this mode to work.

- **Host device_id** is the device_id of remote device. It is optional and applicable only when “redirection” mode is selected. The device id of the device to which redirection is to be done is to be provided here when used along with redirection_server or redirection_client.

4.28.4 Return Value

```
{
        "status": ”string”,
```

"error":"string"
}

4.28.5 Example

Telnet Mode

switch soip: ENABLE baudrate: 115200-8n1 mode: telnet
{
"status":"SUCCESS",
"error":"NULL"
}

Redirection Mode

In 1st device

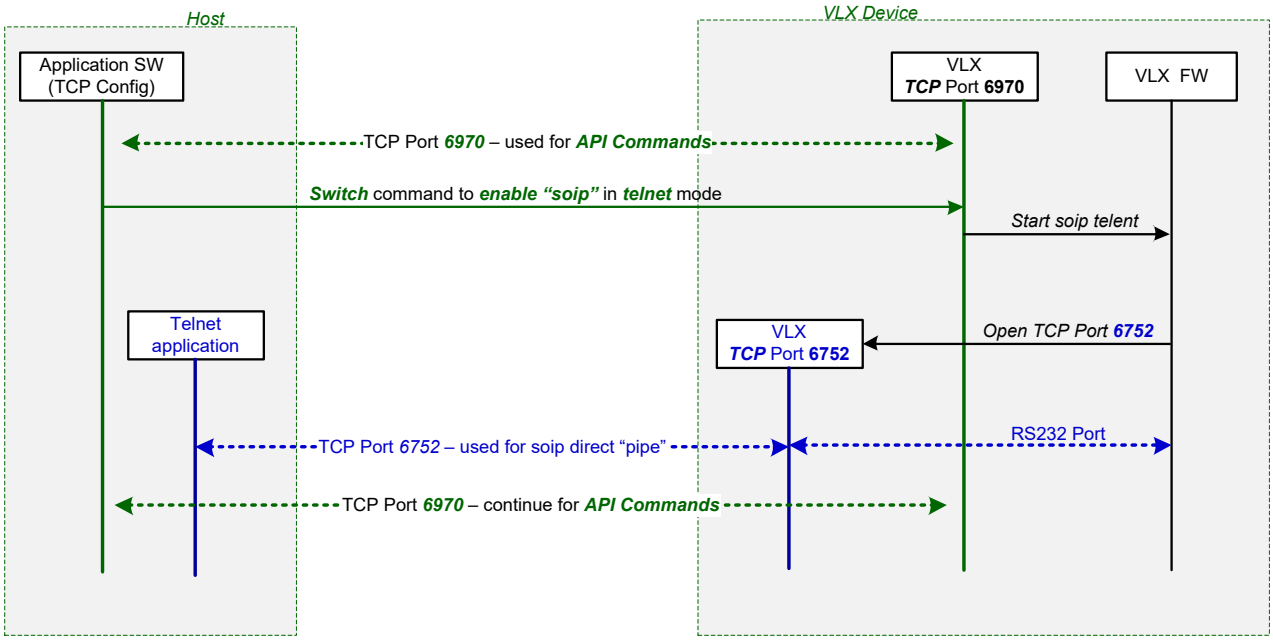
switch soip: ENABLE baudrate: 115200-8n1 mode: redirection_server vlx-series-enc-0004
{
"status":"SUCCESS",
"error":"NULL"
}

In 2nd device

switch soip: ENABLE baudrate: 115200-8n1 mode: redirection_client vlx-series-dec-0018
{
"status":"SUCCESS",
"error":"NULL"
}

switch soip: DISABLE
{
"status":"SUCCESS",
"error":"NULL"
}

Note: Figure below describe TCP Port connection for VLX commands and the additional TCP Port created in VLX for "soip" direct connection (using the above "switch" command with "telnet" option.


Host
VLX Device
Application SW
(TCP Config)
VLX
TCP Port 6970
VLX FW
TCP Port 6970 – used for API Commands
Switch command to enable "soip" in telnet mode
Start soip telent
Telnet
application
VLX
TCP Port 6752
Open TCP Port 6752
RS232 Port
TCP Port 6752 – used for soip direct "pipe"
TCP Port 6970 – continue for API Commands

5 Appendix A

The following sections illustrate different forms/examples of the get command.

5.1 get status

{"status":"SUCCESS","result":"s_idle","error":"NULL"}

5.2 get settings

Decoder

{"status":"SUCCESS","settings":{"mac":"001102F10024","device_id":"vlx-series-dec-0018","FW_version":"3.9.2","display_mode":"video_wall","general_settings":{"vlx_model":"bx","auto_sense":"n","debug":"n","front_panel_lock":"n","serial_port":"","bitmap_files":"","bitmap_usage":"4955,14,4685,0%","chmap_enable":"false","chmap_ir_en":"0"},"hdmi":{"tx":"n","hdcp":"On","local_display_source":"STREAM","stream_source":"","hdcp_capability":""},"ip":{"mode":"autoip","address":"169.254.3.157","mask":"255.255.0.0","gateway":"169.254.0.254","ip_mode":"autoip"},"streams":{"video":{"ip":"225.0.102.4","status":"streaming","wallsize_r":"10","wallsize_c":"10","row_idx":"10","col_idx":"0","overall_width":"1","view_width":"1","overall_height":"1","view_height":"1","h_scale":"0","v_scale":"0","h_shift":"0","v_shift":"0","rotate":"0","frame_rate":"","bit_rate":"","h_size":"1920","v_size":"1080","fps":"60","enable":"true","colour_space":"","bits_per_pixel":"24","scan_mode":"Progressive","source_stable":"true","src1_audio_input":"","src2_audio_input":"","hdmi_audio_output":"none","remote_hostname":"vlx-series-enc-0204",
"no_signal_timeout":"10","turn_off_screen_on_video_lost":"false","scalar_output_mode":"PassThrough"},"rs232":{"soip_status":"off","mode":"","baudrate":"115200","data_bit":"8","stop_bit":"1","parity":"n","connected_device_id":"","soip_connected":"false","soip_ip":"0.0.0.0"}},"nodes":{"analog_audio_input":{"line_in_volume":"5","mute":"off","src1_bitrate_limit":"","src2_bitrate_limit":"","src1_dante_src":"","src2_dante_src":"","dec_dante_src":"line_in"},"analog_audio_output":{"src1_audio_enable":"","src2_audio_enable":"","dec_audio_enable":"false","src1_dante_enable":"","src2_dante_enable":"","dec_dante_enable":"true","src1_line_in_enable":"","src2_line_in_enable":"","dec_line_in_enable":"false","line_out_volume":"5","mute":"off"},"hdmi_monitor":{"connected":"true"},"led":{"status":"normal"}}},"error": "NULL"}

Encoder

{"status":"SUCCESS","settings":{"mac":"001102FDDDFA","device_id":"vlx-series-enc-0204","FW_version":"3.9.2","display_mode":"genlocked","general_settings":{"vlx_model":"wh","auto_sense":"n","debug":"n","front_panel_lock":"n","serial_port":"front","bitmap_files":"","bitmap_usage":"","chmap_enable":"","chmap_ir_en":""},"hdmi":{"tx":"y","hdcp":"On","local_display_source":"IN_PORT1","stream_source":"IN_PORT2","hdcp_capability":"all"},"ip":{"mode":"autoip","address":"169.254.175.83","mask":"255.255.0.0","gateway":"169.254.0.254","ip_mode":"autoip"},"streams":{"video":{"ip":"225.0.102.4","status":"streaming","wallsize_r":"","wallsize_c":"","row_idx":"","col_idx":"","overall_width":"","view_width":"","overall_height":"","view_height":"","h_scale":"","v_scale":"","h_shift":"","v_shift":"","rotate":"","frame_rate":"0","bit_rate":"","h_size":"1920","v_size":"1080","fps":"60","enable":"true","colour_space":"","bits_per_pixel":"24","scan_mode":"Progressive","source_stable":"true","src1_audio_input":"hdmi","src2_audio_input":"hdmi","hdmi_audio_output":"","remot

e_hostname":"","no_signal_timeout":"","turn_off_screen_on_video_lost":"","scalar_output_mode":""},"rs232":{"soip_status":"off","mode":"","baudrate":"115200","data_bit":"8","stop_bit":"1","parity":"n","connected_device_id":"","soip_connected":"false","soip_ip":"0.0.0.0"}},"nodes":{"analog_audio_input":{"line_in_volume":"5","mute":"off","src1_bitrate_limit":"false","src2_bitrate_limit":"false","src1_dante_src":"none","src2_dante_src":"none","dec_dante_src":""},"analog_audio_output":{"src1_audio_enable":"false","src2_audio_enable":"false","dec_audio_enable":"","src1_dante_enable":"false","src2_dante_enable":"false","dec_dante_enable":"","src1_line_in_enable":"false","src2_line_in_enable":"false","dec_line_in_enable":"","line_out_volume":"5","mute":"off"},"hdmi_monitor":{"connected":""},"led":{"status":"normal"}}},"error": "NULL"}

5.3 get local_display_source

Decoder

{"status":"SUCCESS","result":"STREAM","error":"NULL"}

Or

{"status":"SUCCESS","result":"IN_PORT1","error":"NULL"}

Or

{"status":"SUCCESS","result":"IN_PORT2","error":"NULL"}

Encoder

{"status":"SUCCESS","result":"IN_PORT1","error":"NULL"}

Or

{"status":"SUCCESS","result":"IN_PORT1","error":"NULL"}

5.4 get stream_source

{"status":"SUCCESS","result":"IN_PORT1","error":"NULL"}

Or

{"status":"SUCCESS","result":"IN_PORT2","error":"NULL"}

5.5 get dante

{"status":"SUCCESS","result":"hdmi","error":"NULL"}

Or

{"status":"SUCCESS","result":"line_In","error":"NULL"}

Or

{"status":"SUCCESS","result":"none","error":"NULL"}

5.6 get auto_sense

get auto_sense

{"status":"SUCCESS","result":"on","error":"NULL"}

Or

{"status":"SUCCESS","result":"off","error":"NULL"}

5.7 get hdcp_capability

get hdcp_capability

{"status":"SUCCESS","result":"hdcp14","error":"NULL"}

Or

{"status":"SUCCESS","result":"all","error":"NULL"}

Or

{"status":"SUCCESS","result":"none","error":"NULL"}

5.8 get analog_audio_output

{"status":"SUCCESS","result":"on","error":"NULL"}

Or

{"status":"SUCCESS","result":"off","error":"NULL"}

5.9 get bitrate_limit

{"status":"SUCCESS","result":"on","error":"NULL"}

Or

{"status":"SUCCESS","result":"off","error":"NULL"}

5.10 get serial_port_wp

{"status":"SUCCESS","result":"front","error":"NULL"}

Or

{"status":"SUCCESS","result":"rear","error":"NULL"}

5.11 get edid

get edid
{"status":"SUCCESS","result":"0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x17, 0x10, 0x01, 0x09, 0x01, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x01, 0x03, 0x80, 0x73, 0x41, 0x78, 0x2a, 0x7c, 0x11, 0x9e, 0x59, 0x47, 0x9b, 0x27, 0x10, 0x50, 0x54, 0x21, 0x08, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x8c, 0x0a, 0xd0, 0x8a, 0x20, 0xe0, 0x2d, 0x10, 0x10, 0x3e, 0x96, 0x00, 0x04, 0x03, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x45, 0x50, 0x2d, 0x48, 0x44, 0x4d, 0x49, 0x2d, 0x52, 0x58, 0x0a, 0x20, 0x20, 0x00, 0x00, 0x00, 0xfd, 0x00, 0x3b, 0x3d, 0x0f, 0x2e, 0x08, 0x00, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x01, 0xab, 0x02, 0x03, 0x34, 0x71, 0x4a, 0x84, 0x13, 0x05, 0x14, 0x10, 0x02, 0x03, 0x11, 0x12, 0x01, 0x38, 0x09, 0x0f, 0x07, 0x0f, 0x7f, 0x07, 0x15, 0x07, 0x50, 0x3e, 0x06, 0xc0, 0x4d, 0x02, 0x00, 0x57, 0x06, 0x01, 0x5f, 0x7f, 0x01, 0x67, 0x7f, 0x01, 0x83, 0x4f, 0x00, 0x00, 0x67, 0x03, 0x0c, 0x00, 0xff, 0xff, 0x80, 0x1e, 0x01, 0x1d, 0x00, 0x72, 0x51, 0xd0, 0x1e, 0x20, 0x6e, 0x28, 0x55, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x00, 0xbc, 0x52, 0xd0, 0x1e, 0x20, 0xb8, 0x28, 0x55, 0x40, 0x10, 0x09, 0x00, 0x00, 0x00, 0x1e, 0x01, 0x1d, 0x80, 0x18, 0x71, 0x1c, 0x16, 0x20, 0x58, 0x2c, 0x25, 0x00, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x01, 0x1d, 0x80, 0xd0, 0x72, 0x1c, 0x16, 0x20, 0x10, 0x2c, 0x25, 0x80, 0x10, 0x09, 0x00, 0x00, 0x00, 0x9e, 0x00, 0x00, 0x00, 0xd6,","error":"NULL"}

5.12 get remote_hostname

{"status":"SUCCESS","result":"vlx-series-enc-0172","error":"NULL"}

5.13 get ir_auto_pair

{"status":"SUCCESS","result":"y","error":"NULL"}

Or

{"status":"SUCCESS","result":"n","error":"NULL"}

6 Appendix B

The following sections illustrate the Added Response Structures For Events

6.1 Sense Detect

- {"status":"EVENT","device_id":"vlx-series-dec-0018","event_type":"sense_detect","event_details":{"port":"IN_PORT1","value":"plugged"}}

- {"status":"EVENT","device_id":"vlx-series-dec-0018","event_type":"sense_detect","event_details":{"port":"IN_PORT2","value":"unplugged"}}

6.2 Hotplug Detect

- {"status":"EVENT","device_id":"vlx-series-dec-0018","event_type":"hotplug_detect","event_details":{"port":"OUT_PORT1","value":"plugged"}}

- {"status":"EVENT","device_id":"vlx-series-enc-0004","event_type":"hotplug_detect","event_details":{"port":"OUT_PORT1","value":"unplugged"}}

6.3 Source Switch

- {"status":"EVENT","device_id":"vlx-series-enc-0004","event_type":"source_switch","event_details":{"source_type":"local_dsiplay","value":"IN_PORT1"}}

- {"status":"EVENT","device_id":"vlx-series-dec-0018","event_type":"source_switch","event_details":{"source_type":"stream","value":"IN_PORT1"}}

- {"status":"EVENT","device_id":"vlx-series-dec-0018","event_type":"source_switch","event_details":{"source_type":"local_display","value":"IN_PORT2"}}