# HTTP API for FXXX-1XA / HEXK-DCK / FXXK-DCK / DME-10 / HE1080-RPS / Z3CAM / ZCube(-21/SC/SDI) / Z3Dome

For HE4K-01 / HE4K-R4 / MVE-150 / MVE-100R / SME-01 / DME-02/03/04 use Z3-TI-http_api

For firmware version 5.X and higher

## Configuration API Syntax

### Reading Configuration

To read configuration variable, simply perform an HTTP GET to the URL

http://server_address/cgi-bin/control.cgi

with the body as given below.

The reponse will be in JSON format, giving the value of each variable. To read global configuration variables, GET with this body:

```
ctrl=sys&chn=null
```

To read encoder channel configuration variables for channel 1, GET with this body:

```
ctrl=enc&chn=1
```

Python example code for reading configuration:

```
import requests, json, sys

server_url='http://192.168.0.120/cgi-bin/control.cgi'

global_cfg = requests.get(server_url, params='ctrl=sys&chn=null')
print global_cfg.json()

enc_cfg = requests.get(server_url, params='ctrl=enc&chn=1' )
print enc_cfg.json()
```

Sample output:

```
>>> print global_cfg.json()
{u'ipmtu': 1500, u'vid_port': u'microhdmi', u'macaddr':
u'40:cd:3a:04:f0:c3', u'logo_enable': u'off', u'wifi_exists': True,
```

```
u'processor_id': u'QCS605', u'ddns_password': u'', u'model_enable': u'off',
u'enc_current_preset': u'encoder',
u'default_gw': u'192.168.0.1', u'disp_std': u'auto', u'session_id': u'---',
u'ddns_provider': u'freedns',
u'ptp_role': u'auto', u'timezone': u'CST6CDT,M3.2.0,M11.1.0', u'MODEL':
u'Z3-Q603-HD', u'opstate': u'RUNNING',
u'hwversion': u'Z3-Q603-HD', u'ddns_enable': u'off', u'disp_input':
u'primary', u'board_id': u'0xFF10',
u'termserve_remote_enable': u'on', u'hwserial': u'30010110194010011',
u'console_enable': u'on', u'enable_ptp': u'false',
u'camera2_if_type': u'Unused', u'local_netmask': u'255.255.255.0', u'ret':
u'0', u'logo_width': 319, u'MIC_analog_gain_db': 5,
u'camera1_if_type': u'Sony_LVDS', u'opmode': u'encoder', u'syspassword':
u'', u'local_dnsip2': u'0.0.0.0',
u'ddns_hostname': u'', u'use_dhcp': u'0', u'enc_vivpss_mode_enable': u'off',
u'pe3': u'', u'z3_termsrv': u'enabled',
u'MICL_analog_gain_db': -20, u'preview_auto_start': u'1', u'pe7': u'',
u'logo_height': 156,
u'sensor_serial': u'30010310194010011\n', u'ddns_username': u'',
u'eth_speed': u'AUTO', u'nfs_enable': u'off',
u'nfs_server': u'192.168.1.6', u'z3_avmux': u'enabled', u'nmea_enable':
u'off', u'z3_sntp': u'enabled',
u'eth_duplex': u'AUTO', u'z3_fec_enc': u'enabled', u'z3_webproxy':
u'enabled', u'enc_adv_setting': u'off',
u'do_autostart': 1, u'enc_channels': u'1', u'pe8': u'', u'local_dnsip':
u'192.168.0.1', u'local_ip': u'192.168.0.14',
u'pe4': u'', u'pe5': u'', u'pe6': u'', u'diff_serve': 0, u'pe0': u'encoder',
u'pe1': u'', u'pe2': u'',
u'nfs_server_root': u'/c/media', u'enable_snmp': u'false', u'fpgafileglob':
u'', u'enable_sntp': u'true',
u'sysdevicename': u'Z3-Q603-HD', u'sntp_servers': u'pool.ntp.org',
u'zfinder_enable': u'on', u'model_name': u'Z3-Encoder',
u'hw50serial': u'50010110194010011', u'visca_present': u'true'}

print enc_cfg.json()
{u'vmulticastdest': u'225.1.2.3', u'tslowlat': u'on', u'klvmuxmethod':
u'sync', u'srt_destAddr': u'192.168.0.6',
u'vcrop_height': 1080, u'zixifecblock': u'50', u'telopcharsize': u'32',
u'source_status_str': u'+CAMERA 1920x1080p 60.00 fps\n',
u'vratectrl': u'cbr', u'rotate_angle': 0, u'zixisession': u'test',
u'asource': u'CAMERA', u'aport': u'8700', u'pmtpid': 31,
u'authonoff': u'off', u'srt_encrypt': 0, u'vcodec': u'h265', u'klvmode':
u'sdi', u'auxonoff': u'off', u'channel': 1,
u'mounts': u'', u'vcrop_width': 1920, u'vcrop_x': 0, u'vcrop_y': 0,
u'amulticastdest': u'225.1.2.3', u'vprofile': u'high',
u'klvsrc': u'/dev/gv7601.0', u'auth_passwd': u'password', u'teloplocation':
u'top_left', u'gps_overlay_device': u'/dev/ttyAMA0',
u'ret': u'0', u'klvserialbaud': u'115200', u'rtsp_auth_password': u'admin',
u'zixirateadjen': u'on', u'zixiauthen': u'off',
u'vinterlacemode': u'combine', u'vgopsize': 60, u'mmulticastdest':
u'225.1.2.3', u'vcrop_enable': u'off', u'klvenable': u'off',
```

```
u'preset': u'actv_preset', u'mport': u'8800', u'auth_user': u'user',
u'gps_overlay_enable': u'off', u'zixioverhead': u'15',
u'tsrate': u'5000K', u'enc_status': u'RUNNING', u'pcrinterval': 50,
u'aptspcr': 250, u'filesize': u'1024M', u'klvbrate': u'1000',
u'acodec': u'fdk_aaclc', u'lowdelay_opt': u'off', u'apid': 120,
u'zixilatency': u'500', u'rtsp_auth_enable': u'off', u'srt_mode': 0,
u'klvpid': u'35', u'nfstrength': u'0', u'vbitrate': u'4M',
u'rtsp_auth_username': u'admin', u'zixiuser': u'user',
u'gps_overlay_location': u'top_right', u'vprotocol': u'rtsp', u'feconoff':
u'off', u'vdelay': 1000, u'vsource': u'CAMERA',
u'fprefix': u'MOV1_%C', u'telopenable': u'off', u'teloptext': u'Z3-Q603-HD',
u'storage': u'/media/sda1', u'vquality': u'balanced',
u'asamplerate': u'1', u'vgdr': u'off', u'rotate_enable': u'off',
u'avmux_index': u'streaming', u'abitrate': u'128000', u'vpid': 221,
u'feccol': 5, u'gps_overlay_char_size': u'32', u'vframeratediv': 1,
u'pipenable': u'off', u'vres': u'follow_input', u'pcrpid': 521,
u'aenable': u'off', u'apair': 0, u'piplocation': u'top_right',
u'rtmp265_enable': u'off', u'srt_pass': u'password1234',
u'vdest': u'192.168.0.6:8600', u'fecrow': 1}
```


---

# Encoder Channel Configuration

### aenable

Description: Enable audio channel

Possible values: yes, no

### acodec

Description: Audio encoder algorithm selection

Possible values: dsp_aaclc, fdk_aaclc, fdk_aache, fdk_aache_v2

### abitrate

Description: Audio encode bitrate

Possible values: Depends on codec; in units of bits per second

### asamplerate

Description: Audio sample rate

Possible values: 48000, 44100

## asource

Description: Select audio input

Possible values: MICL (microphone, line input levels)

```
MIC    (unpowered microphone levels)
HDMI   (microHDMI input, for HE4K-DCK/FV4K-13A only)
HD-SDI (SDI input, for DME-10/FSDI-DCK/FSDI-13A only)
```

## apid

Description: Audio PID for transport stream.

Possible values: 32 to 8191 - must not conflict with other PID assignments.

## aport

Description: Destination UDP port for RTP audio

## aptspcr

Description: For transport stream modes only – intial PTS to PCR offset for audio, in milliseconds

Possible values: 100 to 2000

## auth_passwd

Description: Password for RTMP authentication.

## auth_user

Description: Username for RTMP authentication.

## authonoff

Description: Enable or disable RTMP authentication. Default: off Possible values: on, off

## channel

Description: Encoder channel number

Possible values: 1, 2, 3

## feccol

Description: Column count for ProMPEG.

Possible values: integer greater than 1

## feconoff

Description: Enable ProMPEG if supported.

Possible values: on, off

## fecrow

Description: Row count for ProMPEG.

Possible values: integer greater than 1

## filesize

Description: Size of TSFILE or AUX file recording, in bytes.

A “K” suffix indicates kilobytes (thousands of bytes/second).

A “M” suffix indicates megabytes (millions of bytes/second).

## fprefix

Description: Relative filename for TSFILE or AUX file recording. Note that “fprefix” does NOT include the absolute media path – the “storage” parameter is the proper place to specify the absolute media path.

## storage

Absolute path to mounted media device (e.g. /media/sda1/)

## avmux_index

For MP4 recording only. Possible values: normal, streaming

normal means MOOV atom at the end of file.

streaming means MOOF atoms interspersed throughout the file, which is more tolerant of errors and truncated files.

streaming is recommended for most recording scenarios. Only select “normal” if you want conventional MOOV atom at the end of file, if some legacy tool or decoder requires it.

## telopcharsize

OSD overlay text character size in pixels

Possible values: 16, 32, or 64

## telopenable

Enable or disable telop (i.e. OSD overlay of text).

Possible values: on, off

## teloplocation

Where to overlay text in the encoded video

Possible values: top_left, top_right, top_center, bottom_left, bottom_right, bottom_center.

## teloptext

Text string for OSD overlay

## telopfont (Qualcomm Only)

The font to be used to overlay text in the encoded video

Possible values: Universal or Monospace

## vgopsize

Description: Distance between I frames (key frames) in GOP sequence

Possible values: 1 through 240

## vbitrate

Description: Video bitrate in bits per second.

Possible values: A “K” suffix indicates kilobits (thousands of bits/second).

A “M” suffix indicates megabits (millions of bits/second).

Note: In UDP transport stream case, the tsrate should be set

higher than the vbitrate, with at least 15% margin.

## vdelay

Description: Video maximum burst size in milliseconds.

Possible values: 100 to 2000

## vdest

Description: Destination URL address for encoded bitstream. May be interpreted differently depending on “vprotocol” setting.

For UDP and RTP transports:

client_ip:client_port

For RTMP transport:

```
server_ip[:server_port]/application/streamname
```

## vframeratediv

Description: Divide video input frame rate by this number to get encode frame rate. The encoder will discard (frameratediv-1) out of (frameratediv) frames.

Possible values: 1, 2, 3, 4, 5, 6

## vgdr

Description: Enable or disable Gradual Data Refresh. Reduces I-frame size (reduces latency). At lower bitrates, a horizontal rolling artifact may be visible.

Possible values: on, off

### vcodec

Description: Specifiec which codec to encode with.

Possible values: h265, h264, mjpeg

### vprofile

Description: H.264 profile (see https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC#Profiles for full description)

Possible values: baseline, main, high

### vprotocol

Description: Protocol used to transport the encoded bitstream.

Possible values: rtp, rtmp, udp, asi, mts, tsrtp

### vquality

Description: Set balance between low latency, and higher quality+bitrate compliance.

Possible values: lowlat, balanced, high

### vratectrl

Description: Rate control mode for encoder.

Possible values: cbr, vbr

vbr mode may be burstier than cbr.

### vres

Description: Video resolution

Possible values: follow_input preserve video source resolution, no resize

```
WxH           resize to width W and height H
```

## vsource

Description: Video input from which encoder will source its video

Possible values: Board-dependent

## pipenable

Description: Enable PIP of channel 2 on channel 1 encode. Only available on FV4K/FV2K/FSDI

Possible values: on, off

## piplocaton

Description: Sets the location of the PIP on channel one. Only available on FV4K/FV2K/FSDI

Possible values: top_left, top_right, bottom_left, bottom_right

## vinterlacemode

Description: Sets the interlaced mode for the video input

Possible values: combined, separated (H265 only)

## vmulticastdest

Description: video multicast destination for RTSP. Used by ONVIF StartMulticast/StopMulticast functions

Possible values: valid multicast addresses. Ranges are 224.0.0.0 to 239.255.255.255

## amulticastdest

Description: audio multicast destination for RTSP. Used by ONVIF StartMulticast/StopMulticast functions

Possible values: valid multicast addresses. The ranges are 224.0.0.0 to 239.255.255.255

## rtsp_auth_enable

Description: enables/disables rtsp authorization for stream

Possible values: on, off

### rtsp_auth_username

Description: username for rtsp authorization

Possible values: alpha numeric string

### rtsp_auth_password

Description: password for rtsp authorization

Possible values: *

## Transport Stream Parameters

The following parameters only apply to transport stream encoding (transport is set to udp, asi, tsfile, mts, or tsrtp).

### vpid

Description: Video PID for transport stream.

Possible values: 32 to 8191 – must not conflict with other PID assignments

### apid

Description: Audio PID for transport stream.

Possible values: 32 to 8191 – must not conflict with other PID assignments

### pcrpid

Description: PCR PID for transport stream.

Possible values: 32 to 8191 – must not conflict with other PID assignments

### pmtpid

Description: PMT PID for transport stream.

Possible values: 16 to 31 – must not conflict with other PID assignments

## tsrate

Description: Transport stream total transport rate in bits per second.

Possible values: The 'K' suffix indicates kilobits (thousands of bits/second). The 'M' suffix indicates megabits (millions of bits/second).

Note: In UDP transport stream case, the tsrate should be set higher than the vbitrate, with at least 15% margin.

## tslowlat

Description: Low latency transport stream mode (VBR).

Possible values: off, on

## auxonoff

Describe: Enable AUXillary file recording in UDP mode – allows simultaneous UDP streaming and TS file recording on the same channel. Only valid if transport is UDP and tslowlat is off

The path to storage is set by the “storage” parameter.

The filename is set by the “fprefix” parameter. The default value is “MOV1_%F_%T”. The “%F” specifier inserts a date stamp, and the “%T” specifier inserts a time stamp.

The recording file size is set by the “filesize” parameter. When the file reaches this size, a new file will be created.

## klvbrate

Description: Bitrate to allocate in transport stream for KLV data.

## klvenable

Description: Enable KLV capture from SDI, UART or debug file, if supported.

Possible values: off, on

## klvmode

Description: Type of KLV source device, whose path is specified in klvsrc

Possible values: sdi, serial, file

### klvserialbaud

Description: If klv_mode is serial, specify the baud rate

Possible values: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200

### klvmuxmethod

Description: Select synchronous or asynchronous transport stream KLV multiplexing, as specified in MISB ST 1402

Possible values: sync, async

### klvpid

Description: PID for KLV data in transport stream

### klvsrc

Description: Source device for KLV metadata

### pcrinterval

Description: Interval between PCR packets in milliseconds.

Possible values: 30 - 100

## Global Configuration

---

## Action Syntax

## Generic Actions (POST)

Python example code for starting/stopping all channels:

```
import requests, json
import sys, getopt

actionname='StartChannel'
```

```
server_url='http://192.168.0.120'

control_cgi_url = server_url + '/cgi-bin/control.cgi'

payload = 'ctrl=sys&chn=null'
sysctrl = requests.get(control_cgi_url, params=payload)

j = sysctrl.json()
print j

enc_channels_string = j['enc_channels']
opmode  = j['opmode']

enc_channels = enc_channels_string.split( ',' )

print 'enc_channels', enc_channels, 'opmode', opmode

channel_list = enc_channels

for channel in channel_list :
# Remove leading C
if channel[0:0] == 'C':
channel = channel[1:]
print actionname, ' ', opmode, 'channel', channel
payload = 'action='+actionname+'&'+ 'chn=' +channel+'&' + 'loadfromdb=true';
headers={'Content-Type':'application/x-www-form-urlencoded; charset=UTF-8'}
r = requests.post(control_cgi_url, data=payload, headers=headers)

print r.text
```

The “loadfromdb=true” parameter will load the current settings from the database.

If you omit the loadfromdb parameter, *none* of the database settings will be used – any parameters you do not supply, will be set to default values.

For example, this is what a StartChannel request from the web server looks like:

```
action=StartChannel&chn=1&vsource=&vres=follow_input&vcodec=h265&vgdr=on&vpr
ofile=high&vratectrl=cbr&vbitrate=2M&vframeratediv=1&vgopsize=60&vprotocol=r
tsp&vdest=192.168.0.6:8600&storage=&fprefix=MOV1_%F_%T_&vpid=221&vdelay=1000
&pcrpid=521&pcrinterval=50&pmtpid=31&tsrate=3000K&tslowlat=on&feconoff=off&f
ecrow=1&feccol=5&zixioverhead=15&zixiauthen=off&zixisession=test&zixiuser=us
er&aenable=on&asource=MICL&acodec=fdk_aaclc&abitrate=128000&asamplerate=4800
0&aport=8700&apid=120&aptspcr=250&klvenable=off&klvmode=sdi&klvsrc=/dev/gv76
01.0&klvpid=35&klvbrate=1000&authonoff=off&auth_user=user&auth_passwd=passwo
rd&auxonoff=off&filesize=1024M&telopenable=off&teloptext=ENCDEV&teloplocatio
n=top_left&telopcharsize=32&vquality=balanced
```

## AddChannel

Add a channel. Check the channel number is valid or not. Based on it, update the channel number and preseting values in the Database.

**Parameters:**

***enc_channels*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4

Returns:

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
action=AddChannel&newchn=2
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

**Example output:**

```
Add Channel response: {"ret":"0","status":"OK"}
```

## AddHistory

Get the channel number and URL as an input and update this value along with the index in the database.

**Parameters:**

---

***channel*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***histidx*** Index of the channel.

***url*** URL of the channel.

Returns:

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## camera_inquiry

---

Read vendor and model info from the visca camera.

**Parameters:**

---

***cam_index*** Read the camera index value. Possible values: 1 = Camera 1, 2 = Camera 2.

---

***function***

Returns:

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=camera_inquiry&cam_index=1
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**


```
Camera Inquiry response:
{"auto_focus_limit":45056,"camera_idx":1,"echo_commands":true,"focus_label":
["1cm",null,null,null,null,null,null,null,null,null,null,null,null,null,null
,null,"5cm",null,null,null,null,null,null,null,null,null,null,null,null,null
,null,null,"15cm",null,null,null,null,null,null,null,null,null,null,null,nul
l,null,null,null,"30cm",null,null,null,null,null,null,null,null,null,null,nu
ll,null,null,null,null,"50cm",null,null,null,null,null,null,null,null,null,n
ull,null,null,null,null,null,"1m",null,null,null,null,null,null,null,null,nu
ll,null,null,null,null,null,null,"1.5m",null,null,null,null,null,null,null,n
ull,null,null,null,null,null,null,null,"1.7m",null,null,null,null,null,null,
null,null,null,null,null,null,null,null,null,"2m",null,null,null,null,null,n
ull,null,null,null,null,null,null,null,null,null,"2.5m",null,null,null,null,
null,null,null,null,null,null,null,null,null,null,null,"3.3m",null,null,null
,null,null,null,null,null,null,null,null,null,null,null,null,"5m",null,null,
null,null,null,null,null,null,null,null,null,null,null,null,null,"10m",null,
null,null,null,null,null,null,null,null,null,null,null,null,null,null,"Inf"]
,"gain_label":["0db",null,null,null,null,null,null,null,"24db",null,null,nul
l,null,null,null,null,"48db"],"has_digital_zoom":true,"has_ir_cut_filter":tr
ue,"has_visibility_enhancement":true,"iris_label":["0",null,null,null,null,n
ull,null,null,null,null,null,null,null,null,null,null,"F4.8",null,null,null,null,
null,null,null,null,null,"F2.0"],"lock_fd":16,"max_focus":57344,"max_gain":17,"ma
x_hlc_level":1,"max_hlc_level_mask":1,"max_iris":25,"max_shutter":33,"max_sl
ow_shutter_limit":255,"min_focus":4096,"min_gain":1,"min_iris":0,"min_shutte
r":6,"min_slow_shutter_limit":0,"monitor_mode_table":{"1080p-25":8,"1080p-29
.97":6,"1080p-50":20,"1080p-59.94":19,"2160p-25":30,"2160p-29.97":29,"720p-5
0":12,"720p-59.94":9},"ret":0,"shutter_label":["1/1",null,null,null,null,nul
l,null,null,null,null,null,null,null,null,null,"1/100",null,null,null,null,null,n
ull,null,null,null,null,null,"1/10K"],"visca_digital_zoom_table":[[1,16384],
[1.1325,18304],[1.28,19968],[1.5,21846],[1.6,22528],[1.79,23616],[1.925,2425
6],[2,24576],[3,27307],[4,28672],[5,29492],[6,30038],[7,30428],[8,30720],[9,
30948],[10,31130],[11,31279],[12,31424]],"visca_fw_version_id":"0101","visca
_ignore_not_exec":{"raw":true,"stable_zoom_off":true,"stable_zoom_on":true,"
wb_manual_bgain_direct":true,"wb_manual_rgain_direct":true},"visca_localport
":1000,"visca_model_id":1802,"visca_name":"Sony FCB-
ER8550","visca_nest_count":0,"visca_optical_zoom_table":[[1,4.4,0],[2,8.8,35
21],[3,13.2,6252],[4,17.6,8214],[5,22,9620],[6,26.4,10679],[7,30.8,11515],[8
,35.2,12208],[9,39.6,12812],[10,44,13357],[11,48.4,13832],[12,52.8,14250],[1
3,57.2,14620],[14,61.6,14950],[15,66,15248],[16,70.4,15516],[17,74.8,15761],
```

```
[18,79.2,15986],[19,83.6,16192],[20,88.4,16384]],"visca_vendor_id":32,"visca_zoom_max":31424,"visca_zoom_max_optical":16384,"visca_zoom_min":0}
```


---

## CameraControl

---

Send control commands to camera.

Only one command can be sent at a time; the list below are the only supported VISCA commands through the HTTP API. To use VISCA commands that are not listed, usage of the TCP socket can be used to send those VISCA commands.

API Format:

```
action=CameraControl&interface=Visca&command=visca_command&cam_index=1

interface possible values: Visca, Tamarisk, Tau2, Boson, Genlock (default: Visca)
```

**example:**

Commands without value assignments

```
action=CameraControl&command=dzoom_off&cam_index=1
```

Commands with value assignments

```
action=CameraControl&command=set_monitor_mode 1080p-59.94&cam_index=1
```

The Visca commands use the same syntax as the serial menu “V” command, as shown below

For the most current list of VISCA commands supported press “V” from the Serial/SSH session of the Serial Menu.

```
auto_icr_disable
auto_icr_enable
cam_control_inquiry
cam_custom_recall
cam_custom_reset
cam_custom_set
chroma_get_suppress
chroma_suppress  Set Chroma Suppression  (0=none, 1 to 3=chroma supression strength)
clear_if
color_gain  Set Color gain (0=60%, 14=200%)
color_get_gain
color_get_hue
color_hue  Set Color hue (0=-14 degrees, 14=+14 degrees)
```

```
debug_rx
dzoom_combine_mode
dzoom_direct  D-Zoom Position (0x00 to 0xeb)
* Enabled during Separate mode
dzoom_off
dzoom_on
dzoom_separate_mode
dzoom_stop
dzoom_super_res
dzoom_tele_var  p = 0 (Low) .. 7 (High)
* Enabled during Separate Mode
dzoom_wide_var  p = 0 (Low) .. 7 (High)
* Enabled during Separate Mode
dzoom_x1_max
eflip_off
eflip_on
ext1_func_inquiry
ext2_func_inquiry
ext3_func_inquiry
focus_auto
focus_direct  focus pos
Min. 0x1000 Over Inf
0x2000 10m
0x3000 5m
0x4000 3.3m
0x5000 2.5m
0x6000 2m
0x7000 1.7m
0x8000 1.5m
0x9000 1m
0xa000 50cm
0xb000 30cm
0xc000 15cm
0xd000 6cm
0xe000 1cm
focus_get_mode
focus_get_pos
focus_manual
focus_near_limit  focus near limit
Min. 0x1000 Over Inf
0x2000 10m
0x3000 5m
0x4000 3.3m
0x5000 2.5m
0x6000 2m
0x7000 1.7m
0x8000 1.5m
0x9000 1m
0xa000 50cm
0xb000 30cm
0xc000 15cm
```

```
0xd000 6cm
0xe000 1cm
focus_stop
focus_tele_std
focus_tele_var  p = 0 (Low) .. 7 (High)
focus_toggle
focus_wide_std
focus_wide_var  p (0 (Low) .. 7 (High))
get_low_delay_mode  Get low delay mode: 0 - normal, 1 - low delay
get_monitor_mode  get_monitor_mode: Get video mode of camera output
help
high_sensitivity_off
high_sensitivity_on
icr_mode  modes:
auto
on
off
threshold [0-255]
lens_control_inquiry
lens_get_temp
lr_reverse_off
lr_reverse_on
other_inquiry
power_inquiry
power_off
power_on
register_read  Read internal register.
register_write  Write internal register.
* Note: To make register changes effective, run power_off and power_on
set_address
set_low_delay_mode  Set low delay mode: 0 - normal, 1 - low delay
set_monitor_mode  Set monitor mode.
List of modes: 1080p-25
720p-50
720p-59.94
2160p-25
2160p-29.97
1080p-29.97
1080p-59.94
1080p-50
slow_shutter  modes:
on
off
limit [0-255]
trace  enable (0=off, 1=on)
version_inquiry
wb_auto_mode
wb_autotrace_mode
wb_get_bgain
wb_get_mode
wb_get_mode_name
```

```
wb_get_rgain
wb_indoor_mode
wb_manual_bgain_direct Set Manual WB BGain (0..255)
* Enabled during Manual WB mode
wb_manual_bgain_reset
wb_manual_mode
wb_manual_rgain_direct Set Manual WB RGain (0..255)
* Enabled during Manual WB mode
wb_manual_rgain_reset
wb_onepush_mode
wb_onepush_trigger
wb_outdoor_auto_mode
wb_outdoor_mode
wb_sodium_lamp_auto_mode
wb_sodium_lamp_fixed_mode
wb_sodium_lamp_outdoor_mode
zoom_direct  pos
Min. 0x0000
Max. 0x4000 (In Separate mode)
Max. 0x4000 (In Combine mode and DZoom=Off)
Max. 0x59C0 (In Combine mode, DZoom=Super Resolution Zoom, and monitoring
mode QFHD)
Max. 0x6000h (In Combine mode, DZoom=Super Resolution Zoom, and monitoring
mode FHD or less)
Max. 0x7AC0h (In Combine mode and DZoom=On)
zoom_get_pos
zoom_stop
zoom_tele_std
zoom_tele_var  p = 0 (Low) .. 7 (High)
zoom_wide_std
zoom_wide_var  p (0 (Low) .. 7 (High))
```

Tau2 Specific

```
The Color are done in the FPGA:

OFF:
SetColorPallette+off

RESET:
SetColorPallette+reset

COLORS:
SetColorPallette+blackhot
SetColorPallette+whitehot
SetColorPallette+sepia
SetColorPallette+rainbow
SetColorPallette+rain
SetColorPallette+ironbow2
SetColorPallette+ironbow1
SetColorPallette+icefire
```

```
SetColorPallette+glowbow
SetColorPallette+fusion
SetColorPallette+color2
SetColorPallette+color1
```

With dual VISCA camera supported added we need to tell the API which camera we want to talk to. this is done with cam_index

The values accepted are either 1 or 2

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## ClearHistory

---

Delete the decoder history for the channel number and update in the database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

### Command to execute:

```
action=ClearHistory&chn=2
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
Clear History response: {"ret":"0","status":"OK"}
```

---

## DeleteChannel

---

Clear the channel deatils in the database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=DeleteChannel&chn=2
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
Delete Channel response: {"ret":"0","status":"OK"}
```

---

## DeletePreset

---

Get the row as an input and delete the row information in the database.

**Parameters:**

---

***Opmode*** operation mode. Possible values: Encoder or Decoder

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=DeletePreset&deleterow=2
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
Delete Preset response: {"ret":"0","status":"OK"}
```

---

## DisconnectWifiAP

Disconnect the wifi network and stop the wifi process id.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
action=DisconnectWifiAP&cam_index=1
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

**Example output:**

```
Disconnect WiFi AP response: {"ret":"0","status":"OK"}
```

## Dynamic

Dynamic instance for the channel number and update the variable with the value on the fly.

One variable at a time is supported in the ***action=Dynamic***

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***var*** The dynamic variable, See **Possible Variables**

***val*** The value for the dynamic variable

**Possible Variables:**

---

***vratediv*** Adjusts the frame rate. Val range [1, 60]. Example 60 fps: 1 = 60 fps, 2 = 30 fps, 10 = 6 fps.

***vrate*** Adjusts the bitrate for the given channel. Val range [2, 40960]. Example values: 2M, 100K.

***gop*** Adjust the Group of Pictures (GOP). Val range: [1, 65536].

***crop*** Update the encoder resolution/ROI. Val is of format: Width**x**Height**(**X**,**Y**)**.

***nfstrength*** Set the noise filter strength. Val range: [0, 1408]. Note: HiSilicon only.

***analog_gain*** Adjust the MIC gain. Ambarella Val range: [-12, 59]. HiSilicon valid range: [-97, 30]. Note: Unavailable on Qualcomm.

***telop_text*** Update the text in text overlay. Note: Unavailable on Qualcomm.

***pip_enable*** Enable or disable PIP. Val on = enabled, off = disabled. Note: Unavailable on Qualcomm.

***pip_location*** Update the location of PIP. Val is top_right, top_left, bottom_right, bottom_left. Note: Unavailable on Qualcomm.

***startmulticast*** Start multicast streaming for the encoder channel. Val is not required.

***stopmulticast*** Stop multicast streaming for the encoder channel. Val is not required.

***debug*** Set the debug level for logging. Val is 0 = none, 1 = error, 2 = info, 3 = diagnostics, 4 = codeflow, 5 = dataflow.

***logmpp_{module name}*** Set the log level for a logmpp module. Example Var 'logmpp_all' for all modules. Note: HiSilicon only.

***rotate_enable*** Enable or disable encoder rotation. Val on = enabled, off = disabled. Note: HiSilicon only.

***rotate_mode*** Set the encoder rotation mode. Val is all, inside, typical. Note: HiSilicon only.

***rotate_angle*** Set the angle of encoder roation. Val range: [0, 360]. Note: HiSilicon only.

***rotate_center_offset*** Set the center offset of encoder rotation. Val is of format X**,**Y and in range [-127, 127]. Note: HiSilicon only.

***translate_enable*** Enable or disable encoder translate. Val on = enabled, off = disabled. Note: HiSilicon only.

***translate_offset*** Set the offset of encoder translate. Val is of format X**,**Y and in range [-256, 256].

Note: HiSilicon only.

***vconflict*** Test whether a video source conflicts with the FPGA configuration. Note: Does not apply to Ambarella or Qualcomm.

***timecode*** Set timecode insertion into bitstream or overlay. Val is bitstream, overlay, on (bitstream and overlay). Note: HiSilicon only.

***genlock*** Enable or disable genlock. Val on = enabled, off = disabled. Note: HiSilicon only.

***eject*** Eject (un-mount) media at the media path given in Val. Example Val: /media/sda. Note: Unavailable on Qualcomm.

***errors_method_encoder*** Configure the error handling method for the encoder. Val is none, restart (encoder), reboot (system). Note: Ambarella only. Some errors require a system reboot if configured for restart.

***restart_pipeline*** Restart the encoder pipeline to try and fix encoder issues. Val vgroup ID range [0, 3]. Note: Qualcomm only.

***los_blue_screen*** Enable or disable LOS screen for all encode channels. Val is on or 1 = enabled, off or 0 = disabled. Note: Qualcomm only.

***dmalloc*** Mark or log (since last mark) unfreed heap pointers. Val is mark (to set the mark) or log (to print all unfreed pointers). Note: Qualcomm only.

***dump_instance*** Print debug information for the camera from the encoding channel. Val is channel ID [1,6]. Note: Qualcomm only.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

### Command to execute to adjust the bitrate of a channel:

```
action=Dynamic&chn=1&var=vrate&val=1M
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
Dynamic response: {"ret":"0","status":"+OK"}
```

---

## EjectStorage

---

Safely eject the removable storage media from the device.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***dev*** Removable device mount point

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=EjectStorage&chn=1&dev=/media/mmcblk0p1/
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
Eject Storage response: {"ret":"0","status":"OK"}
```

---

## ErasePresets

Delete the preseting values from the Database.

**Parameters:**

***mode*** Get the operation mode. The default mode is encoder. Possible values : Encoder , Decoder.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
action=ErasePresets
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

**Example output:**

```
Erase Presets response: {"ret":"0","status":"OK"}
```

## ExportPresets

Create a new database and export the preseting values from the currernt database to the new database.

**Parameters:**

---

***mode*** Get the operation mode. The default mode is encoder. Possible values : Encoder , Decoder.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=ExportPresets
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
Export Presets response: {"ret":"0","status":"OK"}
```

---

## FactoryReset

---

Remove the opt/config folder from the device and set up the default ip address, net mask, gateway, primary dnsip, dhcp in the device.

**Parameters:**

---

***reset_ip*** Get the reset value. Possible values: 0, 1.

The value 0 only for factory reset, and value 1 for factory and ip reset.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=FactoryReset&reset_ip=0
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
FactoryReset response: {"ret":"0","status":"OK"}
```

---

## fmt_media

---

Format the removable storage media in FAT filesystem format and mount it.

**Parameters:**

---

***disk_name*** disk name of the removable media format.

***partition_name*** partition name of the disk.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=fmt_media&disk_name=/dev/mmcblk0&partition_name=mmcblk0p1
```

Use the python script **saveUser_DIGEST_ARGPARSE.py** its in this document and replace the given action command for verify.

---

**Example output:**

```
Format media response: {"ret":"0","status":"OK"}
```

---

## GetCameraLink

---

Get the value of color table using the system call and pixel format from the fpga.

**Parameters:**

---

***color_table*** Color table. Possible values: 0 , 1.

***pxl_format*** Pixel format.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
```

```
passwd admin
```

---

**Example output:**

```
Get camera link response:
{"color_table":"-1","pxl_format":"5","ret":"0","status":"OK"}
```

---

## GetCameraTab

---

Get the camera tab index and camera type. Provide the html file

**Parameters:**

---

***tab_index*** Get the camera tab index. Possible values: 1 = Camera 1, 2 = Camera 2.

***type*** Get the camera type. Possible values: Visca,

***html*** Provide the html file. Default html file is visca.html. Possible html files are: visca.html, flexio.html.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Get Camera Tab response: {"html":"<!-- labeled slider plugin for jQuery UI -
->\n<link rel=\"stylesheet\" href=\"jquery.ui.labeledslider.css\">\n<script
src=\"jquery.ui.labeledslider.js\"></script>\n\n<div class=\"wrapper\"
id=\"cam_1_wrapper_div\">\n\n<div class=\"left\">\n<fieldset
id=\"cam_1_cameraPreviewId\">\n\n  <p>\n    <label
for=\"cam_1_preview_image_button\">Preview Image:</label>\n        <input
type=\"button\" id=\"cam_1_preview_image_button\"
name=\"camera_previewImageButton\" value=\"Load\"
onclick=\"visca_image_preview(this)\"  />\n  </p>\n  <p>\n    <label
for=\"cam_1_preview_image_button\">Preview Stream:</label>\n        <input
type=\"button\" id=\"cam_1_preview_stream_button\"
name=\"camera_previewStreamButton\" value=\"Start\"
onclick=\"visca_stream_preview(this)\"  />\n        <input type=\"button\"
id=\"cam_1_preview_stop_button\" name=\"camera_previewStopButton\"
value=\"Stop\"  onclick=\"visca_stream_preview_stop(this)\"  />\n  </p>\n
<div id=\"cam_1_preview_div\" class=\"preview\">\n     <img
id=\"cam_1_preview_image\" draggable=\"false\" class=\"preview_img\"
alt=\"ImagePreview\"\tsrc=\"/z3blank_logo.png\" >\n     <img
id=\"cam_1_preview_rt\" class=\"preview_right\" src=\"/images/ui-
rtarrow_60x140.png\" \n          draggable=\"true\"\n
onmousedown=\"preview_arrow_mousedown(this)\"\n
onmouseup=\"preview_arrow_mouseup(this)\"\n
onmousemove=\"preview_arrow_mouse_move(this)\"          \n
ondragstart=\"preview_arrow_drag_start(event)\"\n
ondragend=\"preview_arrow_drag_end(event)\"\n          title=\"Press to pan
right&#13;&#10;Drag right to control pan speed\"\n
style=\"display:none\">\n     <img id=\"cam_1_preview_lt\"
class=\"preview_left\" src=\"/images/ui-ltarrow_60x140.png\"\n
draggable=\"true\"\n
onmousedown=\"preview_arrow_mousedown(this)\"\n
onmouseup=\"preview_arrow_mouseup(this)\"\n
onmousemove=\"preview_arrow_mouse_move(this)\"\n
ondragstart=\"preview_arrow_drag_start(event)\"\n
ondragend=\"preview_arrow_drag_end(event)\"\n          title=\"Press to pan
left&#13;&#10;Drag left to control pan speed\"\n
style=\"display:none\">\n     <img id=\"cam_1_preview_up\"
class=\"preview_up\" src=\"/images/ui-uparrow_140x60.png\"\n
draggable=\"true\"\n
onmousedown=\"preview_arrow_mousedown(this)\"\n
onmouseup=\"preview_arrow_mouseup(this)\"\n
onmousemove=\"preview_arrow_mouse_move(this)\"\n
ondragstart=\"preview_arrow_drag_start(event)\"\n
ondragend=\"preview_arrow_drag_end(event)\"\n          title=\"Press to tilt
up&#13;&#10;Drag up to control tilt speed\"\n
style=\"display:none\">\n     <img id=\"cam_1_preview_dn\"
class=\"preview_down\" src=\"/images/ui-dnarrow_140x60.png\"\n
draggable=\"true\"\n
onmousedown=\"preview_arrow_mousedown(this)\"\n
```

```
onmouseup=\"preview_arrow_mouseup(this)\"\n
onmousemove=\"preview_arrow_mouse_move(this)\"\n
ondragstart=\"preview_arrow_drag_start(event)\"\n
ondragend=\"preview_arrow_drag_end(event)\"\n          title=\"Press to tilt
down&#13;&#10;Drag down to control tilt speed\"\n
style=\"display:none\">\n  </div>\n \n</fieldset>\n\n<div
id=\"cam_1_cameraCenterCropId\" style=\"display: none\">\n<fieldset>\n
<legend>Camera Center Crop</legend>\n<p>\n  <div id=\"cam_1_center_crop\">\n
<input type=\"radio\" name=\"cam_1_radio-center-crop\" id=\"cam_1_radio-
center-crop-enable\" value=\"enable\">\n            <label
for=\"cam_1_radio-center-crop-enable\">enable</label>\n            <input
type=\"radio\" name=\"cam_1_radio-center-crop\" id=\"cam_1_radio-center-
crop-disable\" value=\"disable\">\n            <label for=\"cam_1_radio-
center-crop-disable\">disable</label>\n
</div>\n</p>\n\n</fieldset>\n</div>\n\n\n<div
id=\"cam_1_cameraFocusId\">\n<fieldset>\n<legend>Camera
Focus</legend>\n\n<div id=\"cam_1_focus_slider_group\">\n<p>\n <label
for=\"cam_1_focus_pos\">Current focus position:</label>\n  <input
type=\"text\" id=\"cam_1_focus_pos\" readonly style=\"border:0;
color:#f6931f; font-weight:bold;\">\n</p>\n\n<p>\n    <label>Focus Step
Size: </label>\n\n    <select style=\"width:40px;\"
id=\"cam_1_focus_step_size\" name=\"focusStepSize\" title=\"Focus Step
Size\">\n      <option value=\"0\">0</option>\n      <option
value=\"1\">1</option>\n      <option value=\"2\">2</option>\n      <option
value=\"3\">3</option>\n      <option value=\"4\">4</option>\n      <option
value=\"5\">5</option>\n      <option value=\"6\">6</option>\n      <option
value=\"7\">7</option>\n    </select>\n\n    <input id=\"cam_1_focus_plus\"
type=\"button\" class=\"plus_minus_button\"
onmousedown=\"visca_continuous_focus(this)\"
onmouseup=\"visca_continuous_focus_end(this)\" value=\"+\">\n    <input
id=\"cam_1_focus_minus\" class=\"plus_minus_button\" type=\"button\"
onmousedown=\"visca_continuous_focus(this)\"
onmouseup=\"visca_continuous_focus_end(this)\" value=\"-\">\n  </p>\n\n
<tr>\n      <td class=\"label\" id=\"cam_1_CameraFocusLabelId\">Focus</td>\n
<td>\n           <div id=\"cam_1_camerafocus\"></div>\n      </td>\n
</tr>\n</div>\n<p>\n  <div id=\"cam_1_focus_manual\">\n            <input
type=\"radio\" name=\"cam_1_radio-focus\" id=\"cam_1_radio-auto-focus\"
value=\"auto\">\n            <label for=\"cam_1_radio-auto-
focus\">auto</label>\n            <input type=\"radio\" name=\"cam_1_radio-
focus\" id=\"cam_1_radio-manual-focus\" value=\"manual\">\n
<label for=\"cam_1_radio-manual-focus\">manual</label>\n
</div>\n</p>\n\n</fieldset>\n</div>\n\n<div id=\"cam_1_cameraExposureId\"
>\n<fieldset>\n<legend>Exposure</legend>\n\n<table>\n  <thead>\n    <tr>\n
<th></th>\n    </tr>\n  </thead>\n  <tbody>\n  <tr>\n    <td class=\"label\"
id=\"cam_1_CameraExposureModeId\">Exposure Mode</td>\n    <td>\n
<select name=\"camera_exposure_mode\" id=\"cam_1_exposureModeId\"
onchange=\"cameraExposureModeChange(this)\"\n              title=\"Set
Exposure Mode\">\n        <option value=\"0\">Auto</option>\n        <option
value=\"10\">Shutter Priority</option>\n        <option value=\"11\">Iris
Priority</option>\n        <option value=\"3\">Manual</option>\n
</select>\n    </td>\n    <td class=\"space_10pct\"></td>\n  </tr>\n  <tr
```

id=\"cam_1_shutter_slider_group\" style=\"display: none\">\n <td class=\"label\" id=\"CameraShutterId\">Shutter Speed</td>\n <td id=\"CamerShutterLegend\">\n <div id=\"cam_1_camerashutter\"> \n </td>\n <td></td>\n </tr>\n <tr id=cam_1_iris_slider_group style=\"display: none\">\n <td class=\"label\" id=\"CameraIrisId\">Iris</td>\n <td id=\"CamerIrisLegend\">\n <div id=\"cam_1_camerairis\"> \n </td>\n </tr>\n <tr id=cam_1_gain_slider_group style=\"display: none\">\n <td class=\"label\" id=\"CameraGainId\">Gain</td>\n <td id=\"CameraGainLegend\">\n <div id=\"cam_1_cameragain\"> \n </td>\n <td></td>\n </tr>\n <tr id=\"cam_1_high_sensitivity_id\">\n <td class=\"label\">High Sensitivity:</td>\n <td>\n <input id=\"cam_1_high_sensitivity_checkbox\" type=\"checkbox\" name=\"cam_1_high_sensitivity\" onchange=\"checkbox_high_sensitivity(this)\" />\n </td>\n </tr>\n <tr id=\"cam_1_backlight_id\">\n <td class=\"label\">Backlight:</td>\n <td><input id=\"cam_1_backlight_checkbok\" type=\"checkbox\" name=\"cam_1_backlight\" onchange=\"exposure_backlight(this)\" />\n </tr>\n <tr id=\"cam_1_hlc_level_id\">\n <td class=\"label\" >HLC level:</td>\n <td>\n <select id=\"cam_1_hlc_level_sel\" name=\"cam_1_hlc_level\" onchange=\"camera_hlc_level(this)\" >\n <option value=\"0\">Off</option>\n <option value=\"1\">Low</option>\n <option value=\"2\">Mid</option>\n <option value=\"3\">High</option>\n </select>\n </td>\n </tr>\n <tr id=\"cam_1_hlc_level_mask_id\" style=\"display: none\">\n <td class=\"label\" id=\"CameraShutterId\">HLC level mask</td>\n <td id=\"CameraHLCMask\">\n <div id=\"cam_1_camerahlcmask\"> \n </td>\n <td></td>\n </tr>\n <tr id=\"cam_1_slow_shutter_row_id\">\n <td class=\"label\">Auto Slow Shutter:</td>\n <td><input id=\"cam_1_slow_shutter_id\" type=\"checkbox\" name=\"cam_1_slow_shutter\" onchange=\"visca_slow_shutter(this)\" />\n </tr>\n <tr id=\"cam_1_slow_shutter_limit_row_id\">\n <td class=\"label\">Slow Shutter Limit:</td>\n <td>\n <select id=\"cam_1_slow_shutter_limit_id\" name=\"cam_1_slow_shutter_limit\" onchange=\"visca_slow_shutter_limit(this)\">\n <option value=\"1\">1/30</option>\n <option value=\"2\">1/15</option>\n <option value=\"3\">1/8</option>\n <option value=\"4\">1/4</option>\n <option value=\"5\">1/2</option>\n <option value=\"6\">1/1</option>\n </select>\n </td>\n </tr>\n\n \n </tbody>\n</table>\n</fieldset>\n \n<div id=\"cam_1_cameraColorId\" >\n<fieldset>\n<legend>Color Controls</legend>\n\n<table>\n <thead>\n <tr>\n <th></th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <td class=\"label\" id=\"CameraWbModeLabelId\">White Balance Mode</td>\n <td>\n <select name=\"camera_wb_mode\" id=\"cam_1_WbModeId\" onchange=\"cameraWbModeChange(this)\"\n title=\"Set White Balance Mode\">\n\t <option value=\"auto\">Auto</option>\n\t <option value=\"indoor\">Indoor</option>\n\t <option value=\"onepush\" title=\"Point camera to white surface and press trigger button to set white balance\">One-push</option>\n\t <option value=\"autotrace\">Auto Trace</option>\n\t <option value=\"manual\">Manual</option>\n\t <option value=\"sodium_lamp_auto\">Sodium Lamp Auto</option>\n\t <option

```
value=\"sodium_lamp_fixed\">Sodium Lamp Fixed</option>\n\t  <option
value=\"sodium_lamp_outdoor\">Sodium Lamp Outdoor</option>\n
</select>\n      </td>\n    </tr>\n\n    <tr
id=\"cam_1_CameraWbOnepushRowId\">\n      <td class=\"label\"
id=\"CameraWbOnePushLabelId\">Trigger One-push</td>\n      <td>\n
<input type=\"button\" name=\"camera_wbOnePushBtn\" value=\"One Push WB\"
onclick=\"btn_camera_one_push_wb(this)\" />\n      </td>\n    </tr>\n    <tr
id=\"cam_1_CameraWbRgainRowId\">\n      <td class=\"label\">Manual
RGain</td>\n      <td>\n        <input id=\"cam_1_rgain_spinner\" />\n
</td>\n    </tr>\n    <tr  id=\"cam_1_CameraWbBgainRowId\">\n      <td
class=\"label\">Manual BGain</td>\n      <td>\n        <input
id=\"cam_1_bgain_spinner\" />\n      </td>\n    </tr>\n    <tr>\n      <td
class=\"label\"></td>\n      <td>\n      </td>\n    </tr>\n    <tr>\n
<td class=\"label\" id=\"CameraWbRgainId\">Color Gain</td>\n      <td>\n
<div id=\"cam_1_color_gain_slider\"></div>\n      </td>\n    </tr>\n
<tr>\n      <td id=\"cam_1_CameraColorHueLabel\" class=\"label\">Color
Hue</td>\n      <td>\n           <div id=\"cam_1_color_hue_slider\"></div>\n
</td>\n    </tr>\n\n    <tr>\n      <td
id=\"cam_1_CameraColorSuppressLabel\" class=\"label\">Color Suppress</td>\n
<td>\n        \n          <div id=\"cam_1_ColorSuppressId\">\n
<input type=\"radio\" name=\"cam_1_radio-1\" id=\"cam_1_radio-1\"
value=\"0\">\n            <label for=\"cam_1_radio-1\">off</label>\n
<input type=\"radio\" name=\"cam_1_radio-1\" id=\"cam_1_radio-2\"
value=\"1\">\n            <label for=\"cam_1_radio-2\">1</label>\n
<input type=\"radio\" name=\"cam_1_radio-1\" id=\"cam_1_radio-3\"
value=\"2\">\n            <label for=\"cam_1_radio-3\">2</label>\n
<input type=\"radio\" name=\"cam_1_radio-1\" id=\"cam_1_radio-4\"
value=\"3\">\n            <label for=\"cam_1_radio-4\">3</label>\n
</div>\n      </td>\n    </tr>\n
</tbody>\n</table>\n</fieldset>\n</div>\n\n<div
id=\"cam_1_cameraVisibilityEnhancementId\" style=\"display:none\">\n
<fieldset>\n    <legend>Visibility Enhancement</legend>\n    <table>\n
<tr>\n        <td class=\"label\">Mode</td>\n        <td>\n          <select
style=\"width:130px;\" id=\"cam_1_IdVeMode\" name=\"cam_1_ve_mode\"
title=\"Set Visibility Enhancement mode\"
onchange=\"visca_visibility_enhancement(this)\">\n            <option
value=\"3\">Off</option>\n            <option value=\"6\">Visibility
Enhance</option>\n            <option value=\"2\">Wide Dynamic
Range</option>\n          </select>\n        </td>\n        <td
id=\"cam_1_IdVeParamRow1\" style=\"display:none\" >\n          <p>\n
<label>Display Brightness: </label>\n            <select
style=\"width:50px;\" id=\"cam_1_VeDisplayBrightness\"
name=\"cam_1_ve_brightness\" title=\"Display Brightness\"
onchange=\"visca_visibility_enhancement_param(this)\">           \n
<option value=\"0\">0 (Dark)</option>\n              <option
value=\"1\">1</option>\n              <option value=\"2\">2</option>\n
<option value=\"3\">3</option>\n              <option
value=\"4\">4</option>\n              <option value=\"5\">5</option>\n
<option value=\"6\">6 (Bright)</option>\n            </select>\n
</p>\n        </td>\n        \n      </tr>\n      <tr
id=\"cam_1_IdVeParamRow2\" style=\"display:none\">\n        <td
```

```
class=\"label\">Compensation:</td>\n        <td>\n          <select
style=\"width:120px;\" id=\"cam_1_VeCompensationType\"
name=\"cam_1_ve_compensation_type\" title=\"Brightness compensation\"
onchange=\"visca_visibility_enhancement_param(this)\">           \n
<option value=\"0\">Very Dark</option>\n            <option
value=\"1\">Dark</option>\n            <option
value=\"2\">Standard</option>\n            <option
value=\"3\">Bright</option>\n          </select>\n        </td>\n        <td
>\n          <p>\n            <label>Level: </label>\n            <select
style=\"width:80px;\" id=\"cam_1_VeCompensationLevel\"
name=\"cam_1_ve_compensation_level\" title=\"Compensation Level\"
onchange=\"visca_visibility_enhancement_param(this)\">           \n
<option value=\"0\">Low</option>\n              <option
value=\"1\">Mid</option>\n              <option value=\"2\">High</option>\n
</select>\n            </p>\n        </td>       \n      </tr>\n
</table>\n  </fieldset>\n</div>\n\n</div>\n\n<div class=\"right\">\n<div
id=\"cameraAboutId\" >\n<fieldset>\n  <legend>About</legend>\n\n<table>\n
<thead>\n    <tr>\n      <th></th>\n    </tr>\n  </thead>\n  <tbody>\n
<tr>\n      <td class=\"label\">Camera Model:</td>\n      <td
id=\"cam_1_IdCameraModel\"></td>\n    </tr>\n    <tr>\n      <td
class=\"label\">Version:</td>\n      <td
id=\"cam_1_IdCameraVersion\"></td>\n    </tr>\n    <tr>\n      <td> </td>\n
<td id=\"cam_1_IdCameraFWUpdate\" style=\"display: none\">\n         <input
type=\"button\" id=\"cam_1_viscaFWUpdateButton\"
name=\"cam_1_visca_fw_update_button\" value=\"Update Firmware\"\n
onclick=\"window.location='/visca_fw_upload.html'\"/>\n      </td>\n
</tr>\n\n    <!-- VISCA local port setting moved to term tab\n    <tr
id=\"cameraViscaPortId\">\n      <td class=\"label\">VISCA Control TCP
Port:</td>\n      <td>\n         <input type=\"text\"
name=\"visca_localport\" id=\"visca_LocalPort\"> \n         <input
type=\"button\" id=\"viscaLocalportButton\" name=\"visca_local_port_button\"
value=\"Set\"  onclick=\"cameraUpdateViscaPort(this)\"  />\n      </td>\n
</tr>\n    -->\n    \n  </tbody>\n</table>\n</fieldset>\n</div>\n<div
id=\"cam_1_ptzTabsId\">\n<fieldset>\n<legend>PTZ</legend>\n  <div
id=\"cam_1_tabs\">\n  <ul>\n    <li><a
href=\"#cam_1_tabs-1\">Presets</a></li>\n    <li><a
href=\"#cam_1_tabs-2\">Tours</a></li>\n  </ul>\n  <div
id=\"cam_1_tabs-1\">\n    <!-- Feedback message zone -->\n    <div
id=\"cam_1_PtzPreset_message\"></div>\n    \n    <!-- Grid contents -->\n
<div id=\"cam_1_PtzPreset_tablecontent\"></div>\n    \n    <!-- Paginator
control -->\n    <div id=\"cam_1_PtzPreset_paginator\"></div>\n\n    <div
id=\"cam_1_PtzPreset_buttons\"></div>\n    <input type=\"button\"
id=\"cam_1_ptz_preset_add_button\" name =\"cam_1_ptz_presetAddButton\"
value=\"Add\" onclick=\"preview_ptz_preset_add(this)\" />\n    <input
type=\"button\" id=\"cam_1_ptz_preset_save_button\" name
=\"cam_1_ptz_presetSaveButton\" value=\"Save\"
onclick=\"preview_ptz_preset_save(this)\" />\n    \n  </div>\n  <div
id=\"cam_1_tabs-2\">\n    \n    <select id=\"cam_1_PtzTourSelect\"
onchange=\"preview_ptz_tour_select_change(this)\"> </select>\n    \n    <!--
Feedback message zone -->\n    <div id=\"cam_1_PtzTour_message\"></div>\n
\n    <!-- Grid contents -->\n    <div
```

id=\"cam_1_PtzTour_tablecontent\"> \n \n <!-- Paginator control -->\n <div id=\"cam_1_PtzTour_paginator\"> \n\n <div id=\"cam_1_PtzTour_buttons\"> \n <input type=\"button\" id=\"cam_1_ptz_tour_add_button\" name =\"cam_1_ptz_tourAddButton\" value=\"Add\" onclick=\"preview_ptz_tour_add(this)\" />\n <input type=\"button\" id=\"cam_1_ptz_tour_save_button\" name =\"cam_1_ptz_tourSaveButton\" value=\"Save\" onclick=\"preview_ptz_tour_save(this)\" />\n   \n <input type=\"button\" id=\"cam_1_ptz_tour_start_button\" name =\"cam_1_ptz_tourStartButton\" value=\"Run\" onclick=\"preview_ptz_tour_start(this)\" />\n \n \n \n \n \n<div id=\"cam_1_cameraZoomId\" >\n<fieldset>\n<legend>Camera Zoom</legend>\n \n <label for=\"cam_1_amount\">Current zoom position:</label>\n <input type=\"text\" id=\"cam_1_amount\" readonly style=\"border:0; color:#f6931f; font-weight:bold;\">\n <!--<label for=\"cam_1_amount\" style=\"border:0; color:#f6931f; font-weight:bold;\">%</label>-->\n \n\n \n <label>Zoom Step Size: </label>\n <select style=\"width:40px;\" id=\"cam_1_zoom_step_size\" name=\"zoomStepSize\" title=\"Zoom Step Size\">\n <option value=\"1\">1</option>\n <option value=\"2\">2</option>\n <option value=\"3\">3</option>\n <option value=\"4\">4</option>\n <option value=\"5\">5</option>\n <option value=\"6\">6</option>\n <option value=\"7\">7</option>\n </select>\n\n <input id=\"cam_1_zoom_plus\" class=\"plus_minus_button\" type=\"button\" onmousedown=\"visca_continuous_zoom(this)\" onmouseup=\"visca_continuous_zoom_end(this)\" value=\"+\">\n <input id=\"cam_1_zoom_minus\" class=\"plus_minus_button\" type=\"button\" onmousedown=\"visca_continuous_zoom(this)\" onmouseup=\"visca_continuous_zoom_end(this)\" value=\"-\">\n\n \n\n<div id=\"cam_1_zoom_legend\">\n <div id=\"cam_1_camerazoom\" style=\"width: 45%; margin:15px;\"> \n \n\n<div class=\"zoomleft\">\n \n Wide\n \n \n<div class=\"zoomright\">\n \n Telephoto\n \n \n<div class=\"zoomcheck\">\n<table>\n <thead>\n <tr>\n <th></th>\n </tr>\n </thead>\n <tbody>\n <tr id=\"cam_1_zoom_optical_only\">\n <td class=\"label\">Optical only:</td>\n <td>\n <input id=\"cam_1_zoom_optical_checkbox\" type=\"checkbox\" name=\"cam_1_zoom_optical\" onchange=\"checkbox_zoom_optical(this)\" />\n </td>\n <td class=\"space_60pct\"></td>\n <td></td>\n </tr>\n <tr id=\"cam_1_zoom_stabilize_div\">\n <td class=\"label\" >StableZoom:</td>\n <td>\n <input id=\"cam_1_zoom_stabilize_checkbox\" type=\"checkbox\" name=\"cam_1_zoom_stabilize\" onchange=\"checkbox_zoom_stabilize(this)\" />\n </td>\n <td class=\"space_60pct\"></td>\n <td></td>\n </tr>\n </tbody>\n</table>\n \n\n</fieldset>\n \n<div id=\"cam_1_cameraLatencyId\" >\n<fieldset>\n <legend>Camera Latency</legend>\n\n<table>\n <thead>\n <tr>\n <th></th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <td class=\"label\">Low Latency Mode:</td>\n <td>\n <input type=\"checkbox\" name=\"latency_mode\" id=\"cam_1_latency_Mode\" \n title=\"When checked places camera in Low Delay Mode. When in Low delay mode

```
Digital zoom, Distortion correction, Image stabilizer, and StableZoom
features are disabled\"\n
onchange=\"checkbox_camera_latency(this)\" />\n      </td>\n      <td
class=\"space_60pct\"></td>\n      <td></td>\n    </tr>\n
</tbody>\n</table>\n</fieldset>\n</div>\n\n<div
id=\"cam_1_cameraResolutionId\" >\n<fieldset>\n<legend>Video
Standard</legend>\n<table>\n  <thead>\n    <tr>\n      <th></th>\n
</tr>\n  </thead>\n  <tbody>\n    <tr id=\"cam_1_monitor_mode_id\">\n
<td class=\"label\">Monitoring Mode:</td>\n      <td>\n         <select
name=\"camera_monitor_mode\" id=\"cam_1_CameraMonitorModeId\" \n
title=\"Sets video standard output by camera\"\n
onchange=\"change_camera_monitor_mode(this)\">\n               <option
value=\"1080p-60\">1080p60</option> \n         </select>\n      </td>\n
<td ></td>\n      <td></td>\n    </tr>\n\n    <tr
id=\"cam_1_cameraExtSyncSourceRow\" style=\"display: none\">\n      <td
class=\"label\">Ext Sync Source:</td>\n      <td>\n         <select
name=\"camera_ext_sync_source\" id=\"cam_1_CameraExtSyncSourceId\" \n
title=\"Sets external sync source for camera\"\n
onchange=\"change_camera_ext_sync_source(this)\">\n                <option
value=\"0\">None</option> \n                <option value=\"2\">Genlock
Input</option> \n         </select>\n      </td>\n      <td ></td>\n
<td></td>\n    </tr>\n\n    <tr id=\"cam_1_cameraExtSyncStatusRow\"
style=\"display: none\">\n      <td class=\"label\">Camera Sync:</td>\n
<td id=\"cam_1_IdGenlockStatus\"></td>\n      <td ></td>\n      <td></td>\n
</tr>\n    \n\n  </tbody>\n</table>\n</fieldset>\n</div>\n\n<div
id=\"cam_1_icr_opt\" style=\"display:none\">\n  <fieldset>\n
<legend>ICR</legend>\n    <table>\n      <tr>\n        <td
class=\"label\">IR Cut-Removable</td>\n        <td>\n          <select
style=\"width:100px;\" id=\"cam_1_ir_cut_removable\"
name=\"cam_1_manual_icr\" title=\"Set ICR Mode\"
onchange=\"visca_icr(this)\">\n            <option
value=\"on\">On</option>\n            <option value=\"off\">Off</option>\n
<option value=\"auto\">Auto</option>\n            <option
value=\"auto_color\">Auto Color</option>\n          </select>\n
</td>\n        <td id=\"cam_1_IdIcrThresholdRow\">\n          <p>\n
<label>On->Off Threshold: </label>\n            <input style=\"width:50px;\"
type=\"text\" style=\"width:60px;\" id=\"cam_1_IdIcrThreshold\"
name=\"cam_1_icr_threshold\" title=\"ICR On->Off Threshold (0-255 for 4K
camera, 0-28 for HD camera)\" onchange=\"visca_icr_threshold(this)\" />
\n          </p>\n        </td>\n      </tr>\n\n    </table>\n
</fieldset>\n</div>\n\n<div id=\"cam_1_cameraOrientationId\" >\n<fieldset>\n
<legend>Camera Orientation</legend>\n\n<table>\n  <thead>\n    <tr>\n
<th></th>\n    </tr>\n  </thead>\n  <tbody>\n    <tr>\n      <td
class=\"label\">E-Flip:</td>\n      <td>\n         <input type=\"checkbox\"
name=\"eflip\" id=\"cam_1_eFlipId\" \n               title=\"When checked
reverses the video output from the\ncamera vertically and horizontally\"\n
onchange=\"checkbox_camera_eflip(this)\" />\n      </td>\n      <td
class=\"space_60pct\"></td>\n      <td></td>\n    </tr>\n    <tr>\n      <td
class=\"label\">LR Reverse(Mirror):</td>\n      <td>\n         <input
type=\"checkbox\" name=\"lrreverse\" id=\"cam_1_LrReverseId\" \n
title=\"When checked reverses the video output from the\ncamera
```

```
horizontally\"\n                onchange=\"checkbox_camera_lrreverse(this)\"
/>\n      </td>\n      <td class=\"space_60pct\"></td>\n      <td></td>\n
</tr>\n  </tbody>\n</table>\n</fieldset>\n</div>\n\n<div
id=\"cam_1_cameraImageOptions\">\n  <fieldset>\n    <legend>Camera Image
Options</legend>\n    <table>\n      <thead><tr><th></th></tr></thead>\n
<tbody>\n        <tr>\n          <td class=\"label_25pct\"  >Image Freeze:
</td>\n          <td>\n            <input type=\"checkbox\"
name=\"imgfreeze\" id=\"cam_1_image_freeze_opt\" \n
title=\"Freezes the camera's current image\"\n
onchange=\"image_freeze(this)\" />\n          </td>\n          <td
id=\"cam_1_IdFlickerReductionColumn1\" style=\"display:none\"
class=\"label\">Flicker Reduction: </td>\n          <td
id=\"cam_1_IdFlickerReductionColumn2\" style=\"display:none\" >\n
<input type=\"checkbox\" name=\"cam_1_flicker_reduction\"
id=\"cam_1_image_flicker_reduction_opt\" \n
title=\"Turn on flicker reduction mode.\"\n
onchange=\"visca_flicker_reduction(this)\" /></td>\n        </tr>\n
\n        <tr id=\"cam_1_image_hr_id\">\n          <td
class=\"label_25pct\">High Resolution Mode: </td>\n          <td>\n
<input type=\"checkbox\" name=\"hrmode\" id=\"cam_1_image_hr_opt\" \n
title=\"Turn on high resolution mode.\"\n
onchange=\"image_hr(this)\" />\n          </td>\n          <td
class=\"space_60pct\"></td>\n          <td> </td>\n        </tr>\n\n
<tr id=\"cam_1_image_stable_id\">\n            <td
class=\"label_35pct\">Image Stabilizer: </td>\n            <td>\n
<input type=\"checkbox\" name=\"imgstabilizer\"
id=\"cam_1_image_stabilizer_opt\" \n                  title=\"Turn on image
stabilization.\"\n                  onchange=\"image_stabilizer(this)\" />\n
</td>\n            <td id=\"cam_1_IdStabilizerLevelColumn1\"
style=\"display:none\" class=\"label\">Stablizer Level: </td>\n
<td id=\"cam_1_IdStabilizerLevelColumn2\" style=\"display:none\" >\n
<select style=\"width:80px;\" name=\"imgstabilizer_level\"
id=\"cam_1_image_stabilizer_level_opt\"
onchange=\"image_stabilizer_level(this)\"\n                  title=\"Set
image stabilization level\">\n                <option
value=\"0\">Super</option>\n                <option
value=\"2\">Super+</option>\n              </select>\n            </td>\n
</tr>\n\n          <tr>\n              <td class=\"label_35pct\">Image Black
& White: </td>\n              <td>\n                  <input
type=\"checkbox\" name=\"imgbw\" id=\"cam_1_image_blackwhite_opt\" \n
title=\"Turn on black and white filter.\"\n
onchange=\"image_blackwhite(this)\" />\n              </td>\n
<td class=\"space_60pct\"></td>\n              <td></td>\n
</tr>\n\n      </tbody>\n    </table>\n  </fieldset>
\n</div>\n\n<div id=\"cam_1_camera_nr_div\">\n  <fieldset>\n
<legend>Noise Reduction</legend>\n    <table>\n
<thead><tr><th></th></tr></thead>\n      <tbody>\n          <tr>\n
<td class=\"label\">Noise Reduction: </td>\n              <td>\n
<input type=\"checkbox\" name=\"cam_1_nr_enable\"
id=\"cam_1_image_nr_enable\" \n                    title=\"Enable noise
reduction.&#13;&#10;Recommended setting is enabled&#13;&#10; for lower
```

```
bitrates (<10Mbits/sec)\"\n
onchange=\"video_group_nr_enable(this)\" />\n             </td>\n
<td class=\"space_60pct\"></td>\n              <td></td>\n
</tr>\n            <tr id=\"cam_1_IdNoiseReductionLevelRow\"
style=\"display:none\">\n              <td class=\"label\">2D Level: </td>\n
<td>\n                <select style=\"width:80px;\" name=\"nr_2d_level\"
id=\"cam_1_nr_2d_level_opt\" onchange=\"visca_nr_level(this)\"\n
title=\"Set 2D noise reduction level\">\n                   <option
value=\"0\">0 (off)</option>\n                   <option
value=\"1\">1</option>\n                   <option value=\"2\">2</option>\n
<option value=\"3\">3</option>\n                   <option
value=\"4\">4</option>\n                   <option value=\"5\">5</option>\n
</select>\n              </td>\n              <td class=\"label\">3D Level:
</td>\n              <td>\n                <select style=\"width:80px;\"
name=\"nr_3d_level\" id=\"cam_1_nr_3d_level_opt\"
onchange=\"visca_nr_level(this)\"\n                   title=\"Set 3D noise
reduction level\">\n                   <option value=\"0\">0
(off)</option>\n                   <option value=\"1\">1</option>\n
<option value=\"2\">2</option>\n                   <option
value=\"3\">3</option>\n                   <option value=\"4\">4</option>\n
<option value=\"5\">5</option>\n                </select>\n
</td>\n            </tr>\n      </tbody>\n    </table>\n  </fieldset>
\n</div>\n\n</div>\n</div>\n\n\n","index":"1","status":"OK","type":"visca"}
```


---

## GetStatus

---

Read the encoder status.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***status*** The encoder status can be RUNNING/ STOPPED/ IDLE/ Couldn't deduct input on camera.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Get Status response: {"ret":"0","status":"IDLE"}
```

## GetVideoInputs

Get the sensor id and read the video inputs from the Database.

**Parameters:**

***sensorid*** Sensor ID.

***vport*** video port is Camera source. Possible values: 1 = Camera 1, 2 = Camera 2.

***friendly_name*** name of the video source.

***vdevice*** Video device. Possible values: MIPI, SONYLVDS, LT6911

***input_index*** input device. Possible values: 0, 1.

***bus*** device address. Default NULL

***enabled*** Video input enabled or disabled. Default is 1. Possible values: 0, 1.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Get Video Status response:
{"ret":"0","status":"OK","video_inputs":[{"enabled":"on","friendly_name":"Ca
mera1","input_index":0,"sensorid":"300144","vdevice":"LT6911","vport":"CAMER
A"},{"enabled":"on","friendly_name":"Camera2","input_index":1,"sensorid":"30
0144","vdevice":"SONYLVDS","vport":"CAMERA2"}]}
```

## GetWifiPass

Get the Wifi password.

**Parameters:**

***ssid*** Read the service set identifier

***psk*** Provide the wifi password.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## InsQuery

Querry the PTZ values for the angle.

**Parameters:**

---

***data*** Read the json table and retrieve the index and querry type. Possible querry are YawAngle, PitchAngle, RollAngle, All.

***YawAngle*** provide the yaw angle value from the PTZ.

***PitchAngle*** provide the pitch angle value from the PTZ.

***RollAngle*** provide the roll angle value from the PTZ.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## LoadUser

---

Load the active preset value in the preset table and update in the state table database.

**Parameters:**

---

***enc_current_preset*** Get the preset row and check the opermation mode. The operation modes are encoder / decoder.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## Login

Set the password for the login user.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Login response: {"ret":"0","status":"OK"}
```

## Logout

Logout from the system and update in the Database.

**Parameters:**

***Authorization*** Get the permission level.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Logout response: {"ret":"0","status":"OK"}
```

---

## Overlay

---

Display text or a PNG overlayed on the video feed for a given Channel.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***rgn_idx*** *(Ambarella Specific)* Region Index, index to map overlay. The region index value is system-wide and can be in the range of 1 to 16. Ambarella provides 16 total color look-up tables for Overlay (for all Channels), rgn_idx signifies which color look-up table(CLUT) is in use.

```
 TEXT type=text source=”Channel 1” instance=1 rgn_idx=1
      type=text source=”Channel 2” instance=2 rgn_idx=1
```

In the above example, identical region indices are used, effectively reusing the CLUT. The text will be correct on each channel. When changing the color of the text, when using identical region indices, the last Overlay command issued will override all Overlay instances that are using the same region index, the text will still be different but the color will be the same because they are using the same shared

rgn_idx.

```
PNG type=png source=”/opt/config/images/image.png” instance=1 rgn_idx=2
    type=png source=”/opt/config/images/image.png” instance=2 rgn_idx=2
```

PNG overlay will work in the same way as text overlay when re-using region indices, but this is only helpful when overlaying identical PNGs.
Using different PNGs would cause colors to go wonky due to differing CLUTs for the different PNGs.

The encoder writes the entire CLUT at the rgn_idx with each overlay command.

***rgn_idx*** *(*Qualcomm and DCK*)* Index to map overlay. The region index value can be in the range of 0 to 128, per channel.

***type*** Type of overlay to be used possible values: text or png.

***source*** In the case of text overlay this would be the source text for png overlay this is the path to the png on the encoder. Images must be uploaded to board to be used.

***location*** This is the location of the overlay. Possible values: 'top_left', 'top_right', 'top_center', 'bottom_left', 'bottom_right', 'bottom_center', 'x,y' (negative numbers not supported for x or y).

***char_size*** For text overlay this is the character size. Possible values: 16,32,64, 128.

***layer*** This will set the layer for the overly higher numbers and will overlay over lower numbers if overlapping. The range of layer values are 0 to 7.

***text_color*** Text color, which is in RGB values with Alpha

***outline_color*** Outline color of the text which is in RGB values with Alpha

***outline_enable*** Whether outline is enabled or disabled. Possible values: on, off

***outline_stroke*** The width of the outline stroke is in the range of 1, 2 and 4. If the font sizes value are less than 64 then outline stroke value as 1, if the font size value is equal to 64 then outline stroke value is 2 and if the font size value is more than 64 then outline stroke value is 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 z3overlay.py -ip 192.168.10.127 -username admin -passwd admin -aim -
track -moon -rtsp
```

---

**Example output:**

```
Namespace(aim=True, dbg=False, ip='192.168.10.127', moon=True,
passwd='admin', rtsp=True, track=True, username='admin', uw=False)
Getting Video Stream Dimensions
Video Stream Dimensions:3840x2160
Weather Read From File in 171  Position:top_left MSG:1
{"ret":"0","status":"!Invalid rgn_idx 53\r\n"} AimTrack:[574, 2914] - [2060,
1628]
```

---

**Another method to execute:**

```
curl -s --connect-timeout 30 --max-time 60 --data
"action=Overlay&chn=1&rgn_idx=1&source=/opt/z3/images/fmoon-96.png&type=png&
location=top_center" "http://192.168.10.127/cgi-bin/control.cgi"
```

---

**Example output:**

```
{"ret":"0","status":"+OK"}
```

## OverlayStop

---

Stop overlay.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***rgn_idx*** Index of overly for this channel.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

```
python3 z3overlay.py -ip 192.168.10.127 -username admin -passwd admin -aim -track -moon -rtsp
```

**Another method to execute:**

```
curl -s --connect-timeout 30 --max-time 60 --data "action=OverlayStop&chn=1&rgn_idx=1&source=welcome&type=text&location=top_center" "http://192.168.10.127/cgi-bin/control.cgi"
```

**Example output:**

```
{"ret":"0","status":"+OK"}
```

## PtzAbsoluteMove

Send absolute move command to the PTZ.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzAuxiliary

---

Send PTZ auxiliary value to the PTZ.

**Parameters:**

---

***data*** JSON formatted table with parameters:

- **idx** (integer) - PTZ index. Possible values: 0, 1.
- **Value** (integer) - ON/OFF state of the PTZ AUX function. Possible values: 0 - OFF, 1 - ON.
- **AuxId** (integer) - PTZ AUX index. Possible values: 0 to 255.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzClearPreset

---

Remove the Ptz preset values for the user.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=PtzClearPreset&data={"idx":0,"profile_token":"profile1","preset_token":"2"}
```

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd admin
```

---

**Example output:**

```
PtzClearPreset response: {"ret":"0","status":"OK"}
```

---

## PtzContinuousMove

---

Send continous move command to the PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

Please refer the test code and steps to execute is available in this section with different speed and camera selection for both continuous and step mode

Pan to the left at half speed

```
action=PtzContinuousMove&data={"PanTilt":{"x":-0.5,"y":0},"idx":0}
```

Pan to the right at half speed

```
action=PtzContinuousMove&data={"PanTilt":{"x":0.5,"y":0},"idx":0}
```

Tilt down at max speed

```
action=PtzContinuousMove&data={"PanTilt":{"x":0,"y":1},"idx":0}
```

## PtzGotoPreset

---

Send preset value to PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action: "PtzGotoPreset", data: '{"idx":0,"preset_token":"2"}'
```

**Example output:**

```
PtzGotoPreset response: {"ret":"0","status":"OK"}
```

## PtzNewTourSpot

Get the new tour spot and update in the Database.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action=PtzNewTourSpot&data={"idx":0,"profile_token":"profile1","tour_token":
"1","name":"New_TourSpot"}
```

**Example output:**

```
PtzNewTourSpot response: {"ret":"0","status":"OK"}
```

## PtzPosition

Move the Ptz to the input position

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzPreset

Preset the Ptz based on the value.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzQuery

Get the ptz status and degree value from the ptz

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzRemoveTourSpot

Remove the ptz tour spot

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the profile token, preset tour token, row index.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action: "PtzRemoveTourSpot", data:
'{"profile_token":"profile1","tour_token":"1","row_index":0}'
```

**Example output:**

```
PtzRemoveTourSpot response: {"ret":0,"status":"OK"}
```

## PtzSetPreset

Set the preseting value in the PTZ

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action: "PtzSetPreset", data:
'{"idx":0,"profile_token":"profile1","name":"New_Preset"}'
```

---

**Example output:**

```
PtzSetPreset response: {"ret":"0","status":"OK","token":"2"}
```

---

## PtzStartTour

---

Start the PTZ tour

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action: "PtzStartTour", data:
'{"profile_token":"profile1","tour_token":"1","name":"New_TourSpot","idx":0}
'
```

---

**Example output:**

```
PtzStartTour response: {"ret":"0","status":"OK"}
```

---

## PtzStop

---

Stop the PTZ

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action=PtzStop&data={"mode":1,"idx":0}
```

---

**Example output:**

```
PtzStop response: {"ret":"0","status":"OK"}
```

---

## RemoveJobs

---

Delete the schedule jobs from the Database.

**Parameters:**

---

***purgelist*** List of schedule jobs to be removed.

***rowcnt*** Number of rows.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## RestartBoard

---

Read the operation mode. Whether its encoding / decoding. Stop the channel encoding and restart the board

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
RestartBoard response: {"ret":"0","status":"OK"}
```

---

## SaveCamera

---

Saving the camera settings in the Database.

**Parameters:**

---

***zoom_direct_value*** Zoom value. Possible values: 0 (wide) to 0x7ac0 (full zoom)

***white_balance_mode*** White balance mode. Default value is auto. Possible values: auto / manual / indoor/ one-push / auto trace / sodium lamp auto / sodium lamp fixed / sodium lamp outdoor.

***color_gain*** color gain. Default value is 4. Possible values: Range from 0 (60%) to 14 (200%)

***color_hue*** color hue. Default value is 7. Possible values: Range from 0 (60%) to 14 (200%)

***chroma_suppress*** Chroma suppress. Default value is 1. Possible values: 0=none, 1, 2, 3

***wb_manual_rgain_direct*** white balance manual red gain. Default value is 190. Possible values: 0 to 255

***wb_manual_bgain_direct*** white balance manual blue gain. Default value is 190. Possible values: 0 to 255

***optical_zoom_only*** optical zoom only. Default value is 0. Possible values: 0, 1

***focus_direct_value*** focus direct value. Default value is 4096. Possible values: 0x1000 to 0xf000

***manual_focus*** Manual focus. Default value is auto. Possible values: auto, manual

***exposure_mode*** Exposure mode. Default value is 0. Possible values: 0 - Auto, 10 - Shutter Priority, 11 - Iris -Priority, 3 - Manual.

***shutter*** Shutter. Default value is 7.

***iris*** Iris. Default value is 0.

***gain*** Gain. Default value is 0.

***high_sensitivity*** high sensitivity value. Default value is 3. Possible values 0, 1.

***hlc_level*** hlc level. Default value is 0. Possible values in range 0 to 3.

**hlc_level_mask** mask value of hlc level. Default value is 0. Possible values 0, 1.

***stable_zoom*** stable zoom on or off. Default value is off.

***eflip*** Default value is 5. Possible values 0, 1.

***lr_reverse*** Default value is 0. Possible values 0, 1.

***monitor_mode*** Default value is 1080p-59.94.

***genlock_source*** Default value is 2.

***zoom_step_size*** Default value is 4. Possible values in range (1 - 7).

***focus_step_size*** Default value is 1. Possible values in range (0 - 7).

***manual_icr*** Default value is Off. Possible values: On, Off, Auto, Auto Color

***img_freeze*** Default value is 0. Possible values: 0, 1.

***hr_mode*** High resolution mode. Default value is 0. Possible values: 0, 1.

***img_stabilizer*** Default value is 0. Possible values 0, 1.

***img_bw*** Image black white. Default value is 0. Possible values 0, 1.

***nr_2d_level*** Default value is 5. Possible values in range (0 - 5).

***nr_3d_level*** Default value is 4. Possible values in range (0 - 5).

***icr_threshold*** Default value is 14. Possible values in range (0-255 for 4K camera, 0-28 for HD camera).

***slow_shutter*** Default value is 0. Possible values: 0, 1.

***slow_shutter_limit*** Default value is 4. Possible values in range: (1 to 6)

***flicker_reduction*** Default value is 0. Possible values: 0, 1.

***img_stabilizer_level*** Default value is 0. Possible values: 0, 2.

***wide_dynamic_range*** Default value is 3.

***ve_brightness*** Default value is 3. Possible values in range: 0, 6.

***ve_compensation_type*** Default value is 2. Possible values in range: 0, 3.

***ve_compensation_level*** Default value is 1. Possible values in range: 0, 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Save Camera response: {"ret":"0","status":"OK"}
```

---

## SaveCameraBoson

---

Save color pallete settings of boson camera in the Database.

**Parameters:**

---

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Save Camera Boson response: {"ret":"0","status":"OK"}
```

## SaveCameraDRSTamarisk

Save color pallete settings of DRS Tamarisk camera in the Database.

**Parameters:**

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Save Camera DRS Tamarisk response: {"ret":"0","status":"OK"}
```

---

## SaveCameraLink

---

Saving camera link settings in the Database.

**Parameters:**

---

***pxl_format*** Pixel format. Default value is 1.

***color_table*** Color table. Default value is whitehot.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Save Camera Link response: {"ret":"0","status":"OK"}
```

---

## SaveCameraLx

Saving camera Lx settings in the Database.

**Parameters:**

***SONY_ExposureMode*** Exposure Mode. Default value is 32852. Possible values: 32848 (Program Auto), 32849 (Aperture Priority), 32850 (Shutter Priority), 32851 (Manual Exposure), 32852 (Intelligent Auto).

***name*** Name of the camera. Default value is s3ca_1.

***SONY_ShutterSpeed*** Shutter speed. Defualt value is NULL.

***SONY_ISO*** ISO. Defualt value is NULL.

***SONY_ExposureComp*** Exposure comp. Defualt value is NULL.

***SONY_WhiteBalance*** White balance. Defualt value is NULL.

***SONY_ColorTemp*** Color temperature. Defualt value is NULL.

***SONY_APS_C*** APS_C. Defualt value is NULL.

***SONY_NtscPalSelect*** NTSC / PAL selection. Defualt value is NULL.

***SONY_MovieSteadyMode*** Movie steady mode. Defualt value is NULL.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Save Camera Lx response: {"ret":"0","status":"OK"}
```

---

## SaveCameraTau

---

Saving camera TAU settings in the Database.

**Parameters:**

---

***color_table*** color table value .

***is_active*** Active/Deactive. Default value is 0.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveJobs

---

Save the schedule jobs in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the schedule activities.**.**

***rowcnt*** Get the number of row count.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SavePresets

Check the operation mode whether its in encoder or decoder. Save the preseting value of the mode in the Database.

**Parameters:**

***dec_current_preset*** Save the decoder current preseting values.

***enc_current_preset*** Save the encoder current preseting values.

***preset*** Settings for encoder or decoder.

***enc_channels*** Number of channels. Maximum number of channels is 2 for decoder and 4 for encoder.

***opmode*** Operation mode. Default is Encoder. Possible values: Encoder / Decoder.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SavePresets response: {"ret":"0","status":"OK"}
```

---

## SaveUser

---

Save the state, preset, encoder/decoder settings in the Database.

**Parameters:**

---

***enc_current_preset*** Get the encoder preseting values for the current channel.

***dec_current_preset*** Get the decoder preseting values for the current channel.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SaveUser response: {"ret":"0","status":"OK"}
```

---

## SetAdv

---

Save the encoder advance settings in the Database.

**Parameters:**

---

***enc_adv_setting*** Get the encoder advance setting values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetAdv response: {"ret":"0","status":"OK"}
```

---

## SetAPConfig

---

Set the wifi and its password.

**Parameters:**

---

***passwd*** Get the wifi password

***passwden*** Get wifi enable / disable. Default value is disable.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetAPConfig response: {"ret":"0","status":"OK"}
```

---

## SetAudio

---

Save the audio input settings in the Database.

**Parameters:**

---

***aport*** Get the audio port. Default value is MIC. Possible values are MIC , MICL

***analog_gain_db*** Get the analog gain decibal value. Default value is 0. The value ranges are -97 to 30.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetAudio response: {"ret":"0","status":"OK"}
```

## SetCameraLink

Save the color table and pixel format values in the Database and set the color table and pixel format in the fpga.

**Parameters:**

***val*** Get the value for color table and pixel format.

***opt*** The options are color table and pixel format.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Set Camera Link response: {"ret":"0","status":"OK"}
```

## SetConsole

Set the console value as ttyAMA0 or none.

**Parameters:**

***console_enable*** Get console value is enabled / disabled.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Set Console response: {"ret":"0","status":"OK"}
```

## SetDDNSEnable

Set the Dynamic Domain Name System settings in the Database.

**Parameters:**

***ddns_enable*** Get the ddns enable value. Default value is Off. Possible values; (On, Off)

***ddns_provider*** Get the ddns provider. Default value is freedns. Possible values: *(*freedns, freemyip, dyn, domains.google.com, duckdns.org, no-ip.com, tunnelbroker.net, dynv6.com, cloudxnv.net, dnspod.cn, cloudflare.com).

***ddns_username*** Provide the username.

***ddns_password*** Provide the password.

***ddns_hostname*** Hostname for DDNS login.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Set DDNS Enable response: {"ret":"0","status":"OK"}
```

## SetDebugLevel

Set the logging level. The setting is persistent.

**Parameters:**

---

***sysdebuglevel*** Get the system debug level. Possible values 0=None, 1=Error, 2=Information, 3=Diagnostics, 4=Codeflow, 5=Dataflow.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Debug Level response: {"ret":"0","status":"OK"}
```

---

## SetDevName

---

Set the device name in the Database.

**Parameters:**

---

***sysdevicename*** Get the system device name.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Dev Name response: {"ret":"0","status":"OK"}
```

---

## SetDisplay

---

Controls composite output (passthru video from camera).

**Parameters:**

---

***disp_std*** Get display standard for encoder. Default value is Auto.

***disp_input*** Get display input for encoder. Default value is MACRO_DEFAULT_DISP_INPUT.

***save_only*** Possible values: (True / False)

***disp_layout*** Display layout for decoder. Default value is 1×1.

***disp_mode*** Display mode for decoder. Default value is UltraHD.

***disp_std2*** Display standard for decoder. Default value is auto.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Display response: {"ret":"0","status":"OK"}
```

---

## SetDSCP

---

Set the DSCP configuration settings in the Database.

**Parameters:**

---

***diff_serve*** Get the DSCP configuration value. Default value is 0x00. Possible values are ( 0x00, 0xB8, 0x88, 0x90, 0x98, 0x80, 0x68, 0x70, 0x78, 0x60, 0x48, 0x50, 0x58, 0x28, 0x30, 0x38, 0x01).

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetDSCP response: {"ret":"0","status":"OK"}
```

---

## SetFpga

---

Set the fpga settings in the Database.

**Parameters:**

---

***fpgafileglob*** Get fpga value. Possible values are (boson / cvbs / tamarisk / tau2).

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetFpga response: {"ret":"0","status":"OK"}
```

## SetIp

Control networking settings of the Z3 device such as local IPv4 IP address, netmask, etc. Additionally control Encoder Auto-Start after boot-up and enable/disable showing of advanced WebUI settings.

**Parameters:**

***do_autostart*** Control automatic stream start after bootup. Possible values: 1 = do autostart 0 = do not autostart.

***enc_adv_setting*** Control appearance of advanced settings on WebUI. Possible values: off, on.

***ipmtu*** Control Ethernet IP Maximum Transmission Unit (MTU) size in bytes. Possible values: 576 to 1500.

***eth_speed*** Control Gigabit Ethernet auto-negotiation allowed speeds. Possible values: AUTO, AUTO-100, 1000, 100, 10.

***eth_duplex*** Control Ethernet auto-negotiation allowed duplex. Possible values: AUTO, FULL, HALF.

***local_ip*** IPv4 address.

***local_netmask*** IPv4 netmask.

***default_gw*** IPv4 default gateway.

***local_dnsip*** DNS server primary.

***local_dnsip2*** DNS server secondary.

***use_dhcp*** Controls use of DHCP or static IP. Possible values: 1 (DHCP enabled), 0 (DHCP disabled, use static IP).

***local_hostname*** Network hostname.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=SetIp&default_gw=172.29.0.1&do_autostart=1&enc_adv_setting=on&eth_dup
lex=AUTO&eth_speed=AUTO&ipmtu=1500&local_dnsip=8.8.8.8&local_dnsip2=4.4.4.4&
local_hostname=Z3Dome-4M&local_ip=172.29.10.122&local_netmask:
=255.255.0.0&use_dhcp=1
```

**Example output:**

```
SetIp response: {"ret":"0","status":"OK"}
```

## SetLoginLimit

Set the login limit settings in the Database

**Parameters:**

***maxlogin*** mximum number of login limit. Possible value in range (3 to 99 ).

***login_window*** Login window period, the value are considered in seconds. Possible value in range (60 to 32768).

***lockout_interval*** Lockout interval values are mentioned in seconds, which is used when maximum number of login limit is reached. Possible value in range (-1 to 32768)

***login_timeout*** Login timeout

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetLoginLimit response: {"ret":"0","status":"OK"}
```

## SetNFS

Set the NFS settings in the Database.

**Parameters:**

***nfs_enable*** Enable/Disable nfs

***nfs_server*** Get the nfs server ip address. Default is 192.168.1.6

***nfs_server_root*** Get the server root path / location.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetNFS response: {"ret":"0","status":"OK"}
```

---

## SetSNTP

---

Set SNTP (Simple Network Time Protocol) settings in the Database.

**Parameters:**

---

***enable*** Enable/Disable SNTP. Possible value: true, false, ctime, ptp . Default value true.

***servers*** NTP server or list of NTP servers.

***timezone*** Linux TZ database value for Timezone.

***timezone_name*** Name for a time zone. Default is America/Chicago.

***ptprole*** The Role for doing IEEE-1588 PTP Possible value: auto,server,client. Default is auto

***unixtime*** When **ctime** is used in ***enable*** then the current unix time for the time you want to set.

If the **timezone** is changed then a reboot of the Z3 device is required. You can use POST of **action=RestartBoard** after the **SetSNTP** command returns successfully

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetSNTP response: {"ret":"0","status":"OK"}
```

**Timezone to Timezone_name**

| | |
|---|---|
| GMT0 | Africa/Abidjan |
| GMT0 | Africa/Accra |
| EAT-3 | Africa/Addis Ababa |
| CET-1 | Africa/Algiers |
| EAT-3 | Africa/Asmara |
| GMT0 | Africa/Bamako |
| WAT-1 | Africa/Bangui |
| GMT0 | Africa/Banjul |
| GMT0 | Africa/Bissau |
| CAT-2 | Africa/Blantyre |
| WAT-1 | Africa/Brazzaville |
| CAT-2 | Africa/Bujumbura |
| WET0 | Africa/Casablanca |
| CET-1CEST,M3.5.0,M10.5.0/3 | Africa/Ceuta |
| GMT0 | Africa/Conakry |
| GMT0 | Africa/Dakar |
| EAT-3 | Africa/Dar es Salaam |
| EAT-3 | Africa/Djibouti |
| WAT-1 | Africa/Douala |
| WET0 | Africa/El Aaiun |
| GMT0 | Africa/Freetown |
| CAT-2 | Africa/Gaborone |
| CAT-2 | Africa/Harare |
| SAST-2 | Africa/Johannesburg |
| EAT-3 | Africa/Kampala |
| EAT-3 | Africa/Khartoum |
| CAT-2 | Africa/Kigali |
| WAT-1 | Africa/Kinshasa |
| WAT-1 | Africa/Lagos |
| WAT-1 | Africa/Libreville |
| GMT0 | Africa/Lome |
| WAT-1 | Africa/Luanda |
| CAT-2 | Africa/Lubumbashi |
| CAT-2 | Africa/Lusaka |
| WAT-1 | Africa/Malabo |
| CAT-2 | Africa/Maputo |

| | |
|---|---|
| SAST-2 | Africa/Maseru |
| SAST-2 | Africa/Mbabane |
| EAT-3 | Africa/Mogadishu |
| GMT0 | Africa/Monrovia |
| EAT-3 | Africa/Nairobi |
| WAT-1 | Africa/Ndjamena |
| WAT-1 | Africa/Niamey |
| GMT0 | Africa/Nouakchott |
| GMT0 | Africa/Ouagadougou |
| WAT-1 | Africa/Porto-Novo |
| GMT0 | Africa/Sao Tome |
| EET-2 | Africa/Tripoli |
| CET-1 | Africa/Tunis |
| WAT-1WAST,M9.1.0,M4.1.0 | Africa/Windhoek |
| HAST10HADT,M3.2.0,M11.1.0 | America/Adak |
| AKST9AKDT,M3.2.0,M11.1.0 | America/Anchorage |
| AST4 | America/Anguilla |
| AST4 | America/Antigua |
| BRT3 | America/Araguaina |
| ART3 | America/Argentina/Buenos Aires |
| ART3 | America/Argentina/Catamarca |
| ART3 | America/Argentina/Cordoba |
| ART3 | America/Argentina/Jujuy |
| ART3 | America/Argentina/La Rioja |
| ART3 | America/Argentina/Mendoza |
| ART3 | America/Argentina/Rio Gallegos |
| ART3 | America/Argentina/Salta |
| ART3 | America/Argentina/San Juan |
| ART3 | America/Argentina/Tucuman |
| ART3 | America/Argentina/Ushuaia |
| AST4 | America/Aruba |
| PYT4PYST,M10.1.0/0,M4.2.0/0 | America/Asuncion |
| EST5 | America/Atikokan |
| BRT3 | America/Bahia |
| AST4 | America/Barbados |
| BRT3 | America/Belem |
| CST6 | America/Belize |
| AST4 | America/Blanc-Sablon |
| AMT4 | America/Boa Vista |
| COT5 | America/Bogota |
| MST7MDT,M3.2.0,M11.1.0 | America/Boise |
| MST7MDT,M3.2.0,M11.1.0 | America/Cambridge Bay |
| AMT4AMST,M10.3.0/0,M2.3.0/0 | America/Campo Grande |
| CST6CDT,M4.1.0,M10.5.0 | America/Cancun |
| VET4:30 | America/Caracas |
| GFT3 | America/Cayenne |

| | |
|---|---|
| EST5 | America/Cayman |
| CST6CDT,M3.2.0,M11.1.0 | America/Chicago |
| MST7MDT,M4.1.0,M10.5.0 | America/Chihuahua |
| CST6 | America/Costa Rica |
| AMT4AMST,M10.3.0/0,M2.3.0/0 | America/Cuiaba |
| AST4 | America/Curacao |
| GMT0 | America/Danmarkshavn |
| PST8PDT,M3.2.0,M11.1.0 | America/Dawson |
| MST7 | America/Dawson Creek |
| MST7MDT,M3.2.0,M11.1.0 | America/Denver |
| EST5EDT,M3.2.0,M11.1.0 | America/Detroit |
| AST4 | America/Dominica |
| MST7MDT,M3.2.0,M11.1.0 | America/Edmonton |
| AMT4 | America/Eirunepe |
| CST6 | America/El Salvador |
| BRT3 | America/Fortaleza |
| AST4ADT,M3.2.0,M11.1.0 | America/Glace Bay |
| AST4ADT,M3.2.0/0:01,M11.1.0/0:01 | America/Goose Bay |
| EST5EDT,M3.2.0,M11.1.0 | America/Grand Turk |
| AST4 | America/Grenada |
| AST4 | America/Guadeloupe |
| CST6 | America/Guatemala |
| ECT5 | America/Guayaquil |
| GYT4 | America/Guyana |
| AST4ADT,M3.2.0,M11.1.0 | America/Halifax |
| CST5CDT,M3.2.0/0,M10.5.0/1 | America/Havana |
| MST7 | America/Hermosillo |
| EST5EDT,M3.2.0,M11.1.0 | America/Indiana/Indianapolis |
| CST6CDT,M3.2.0,M11.1.0 | America/Indiana/Knox |
| EST5EDT,M3.2.0,M11.1.0 | America/Indiana/Marengo |
| EST5EDT,M3.2.0,M11.1.0 | America/Indiana/Petersburg |
| CST6CDT,M3.2.0,M11.1.0 | America/Indiana/Tell City |
| EST5EDT,M3.2.0,M11.1.0 | America/Indiana/Vevay |
| EST5EDT,M3.2.0,M11.1.0 | America/Indiana/Vincennes |
| EST5EDT,M3.2.0,M11.1.0 | America/Indiana/Winamac |
| MST7MDT,M3.2.0,M11.1.0 | America/Inuvik |
| EST5EDT,M3.2.0,M11.1.0 | America/Iqaluit |
| EST5 | America/Jamaica |
| AKST9AKDT,M3.2.0,M11.1.0 | America/Juneau |
| EST5EDT,M3.2.0,M11.1.0 | America/Kentucky/Louisville |
| EST5EDT,M3.2.0,M11.1.0 | America/Kentucky/Monticello |
| BOT4 | America/La Paz |
| PET5 | America/Lima |
| PST8PDT,M3.2.0,M11.1.0 | America/Los Angeles |
| BRT3 | America/Maceio |
| CST6 | America/Managua |

| | |
|---|---|
| AMT4 | America/Manaus |
| AST4 | America/Marigot |
| AST4 | America/Martinique |
| CST6CDT,M3.2.0,M11.1.0 | America/Matamoros |
| MST7MDT,M4.1.0,M10.5.0 | America/Mazatlan |
| CST6CDT,M3.2.0,M11.1.0 | America/Menominee |
| CST6CDT,M4.1.0,M10.5.0 | America/Merida |
| CST6CDT,M4.1.0,M10.5.0 | America/Mexico City |
| PMST3PMDT,M3.2.0,M11.1.0 | America/Miquelon |
| AST4ADT,M3.2.0,M11.1.0 | America/Moncton |
| CST6CDT,M4.1.0,M10.5.0 | America/Monterrey |
| UYT3UYST,M10.1.0,M3.2.0 | America/Montevideo |
| EST5EDT,M3.2.0,M11.1.0 | America/Montreal |
| AST4 | America/Montserrat |
| EST5EDT,M3.2.0,M11.1.0 | America/Nassau |
| EST5EDT,M3.2.0,M11.1.0 | America/New York |
| EST5EDT,M3.2.0,M11.1.0 | America/Nipigon |
| AKST9AKDT,M3.2.0,M11.1.0 | America/Nome |
| FNT2 | America/Noronha |
| CST6CDT,M3.2.0,M11.1.0 | America/North Dakota/Center |
| CST6CDT,M3.2.0,M11.1.0 | America/North Dakota/New Salem |
| MST7MDT,M3.2.0,M11.1.0 | America/Ojinaga |
| EST5 | America/Panama |
| EST5EDT,M3.2.0,M11.1.0 | America/Pangnirtung |
| SRT3 | America/Paramaribo |
| MST7 | America/Phoenix |
| AST4 | America/Port of Spain |
| EST5 | America/Port-au-Prince |
| AMT4 | America/Porto Velho |
| AST4 | America/Puerto Rico |
| CST6CDT,M3.2.0,M11.1.0 | America/Rainy River |
| CST6CDT,M3.2.0,M11.1.0 | America/Rankin Inlet |
| BRT3 | America/Recife |
| CST6 | America/Regina |
| AMT4 | America/Rio Branco |
| PST8PDT,M4.1.0,M10.5.0 | America/Santa Isabel |
| BRT3 | America/Santarem |
| AST4 | America/Santo Domingo |
| BRT3BRST,M10.3.0/0,M2.3.0/0 | America/Sao Paulo |
| EGT1EGST,M3.5.0/0,M10.5.0/1 | America/Scoresbysund |
| MST7MDT,M3.2.0,M11.1.0 | America/Shiprock |
| AST4 | America/St Barthelemy |
| NST3:30NDT,M3.2.0/0:01,M11.1.0/0:01 | America/St Johns |
| AST4 | America/St Kitts |
| AST4 | America/St Lucia |
| AST4 | America/St Thomas |

| | |
|---|---|
| AST4 | America/St Vincent |
| CST6 | America/Swift Current |
| CST6 | America/Tegucigalpa |
| AST4ADT,M3.2.0,M11.1.0 | America/Thule |
| EST5EDT,M3.2.0,M11.1.0 | America/Thunder Bay |
| PST8PDT,M3.2.0,M11.1.0 | America/Tijuana |
| EST5EDT,M3.2.0,M11.1.0 | America/Toronto |
| AST4 | America/Tortola |
| PST8PDT,M3.2.0,M11.1.0 | America/Vancouver |
| PST8PDT,M3.2.0,M11.1.0 | America/Whitehorse |
| CST6CDT,M3.2.0,M11.1.0 | America/Winnipeg |
| AKST9AKDT,M3.2.0,M11.1.0 | America/Yakutat |
| MST7MDT,M3.2.0,M11.1.0 | America/Yellowknife |
| WST-8 | Antarctica/Casey |
| DAVT-7 | Antarctica/Davis |
| DDUT-10 | Antarctica/DumontDUrville |
| MIST-11 | Antarctica/Macquarie |
| MAWT-5 | Antarctica/Mawson |
| NZST-12NZDT,M9.5.0,M4.1.0/3 | Antarctica/McMurdo |
| ROTT3 | Antarctica/Rothera |
| NZST-12NZDT,M9.5.0,M4.1.0/3 | Antarctica/South Pole |
| SYOT-3 | Antarctica/Syowa |
| VOST-6 | Antarctica/Vostok |
| CET-1CEST,M3.5.0,M10.5.0/3 | Arctic/Longyearbyen |
| AST-3 | Asia/Aden |
| ALMT-6 | Asia/Almaty |
| ANAT-11ANAST,M3.5.0,M10.5.0/3 | Asia/Anadyr |
| AQTT-5 | Asia/Aqtau |
| AQTT-5 | Asia/Aqtobe |
| TMT-5 | Asia/Ashgabat |
| AST-3 | Asia/Baghdad |
| AST-3 | Asia/Bahrain |
| AZT-4AZST,M3.5.0/4,M10.5.0/5 | Asia/Baku |
| ICT-7 | Asia/Bangkok |
| EET-2EEST,M3.5.0/0,M10.5.0/0 | Asia/Beirut |
| KGT-6 | Asia/Bishkek |
| BNT-8 | Asia/Brunei |
| CHOT-8 | Asia/Choibalsan |
| CST-8 | Asia/Chongqing |
| IST-5:30 | Asia/Colombo |
| EET-2EEST,M4.1.5/0,M10.5.5/0 | Asia/Damascus |
| BDT-6 | Asia/Dhaka |
| TLT-9 | Asia/Dili |
| GST-4 | Asia/Dubai |
| TJT-5 | Asia/Dushanbe |
| EET-2EEST,M3.5.6/0:01,M9.1.5 | Asia/Gaza |

| | |
|---|---|
| CST-8 | Asia/Harbin |
| ICT-7 | Asia/Ho Chi Minh |
| HKT-8 | Asia/Hong Kong |
| HOVT-7 | Asia/Hovd |
| IRKT-8IRKST,M3.5.0,M10.5.0/3 | Asia/Irkutsk |
| WIT-7 | Asia/Jakarta |
| EIT-9 | Asia/Jayapura |
| AFT-4:30 | Asia/Kabul |
| PETT-11PETST,M3.5.0,M10.5.0/3 | Asia/Kamchatka |
| PKT-5 | Asia/Karachi |
| CST-8 | Asia/Kashgar |
| NPT-5:45 | Asia/Kathmandu |
| IST-5:30 | Asia/Kolkata |
| KRAT-7KRAST,M3.5.0,M10.5.0/3 | Asia/Krasnoyarsk |
| MYT-8 | Asia/Kuala Lumpur |
| MYT-8 | Asia/Kuching |
| AST-3 | Asia/Kuwait |
| CST-8 | Asia/Macau |
| MAGT-11MAGST,M3.5.0,M10.5.0/3 | Asia/Magadan |
| CIT-8 | Asia/Makassar |
| PHT-8 | Asia/Manila |
| GST-4 | Asia/Muscat |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Asia/Nicosia |
| NOVT-6NOVST,M3.5.0,M10.5.0/3 | Asia/Novokuznetsk |
| NOVT-6NOVST,M3.5.0,M10.5.0/3 | Asia/Novosibirsk |
| OMST-7 | Asia/Omsk |
| ORAT-5 | Asia/Oral |
| ICT-7 | Asia/Phnom Penh |
| WIT-7 | Asia/Pontianak |
| KST-9 | Asia/Pyongyang |
| AST-3 | Asia/Qatar |
| QYZT-6 | Asia/Qyzylorda |
| MMT-6:30 | Asia/Rangoon |
| AST-3 | Asia/Riyadh |
| SAKT-10SAKST,M3.5.0,M10.5.0/3 | Asia/Sakhalin |
| UZT-5 | Asia/Samarkand |
| KST-9 | Asia/Seoul |
| CST-8 | Asia/Shanghai |
| SGT-8 | Asia/Singapore |
| CST-8 | Asia/Taipei |
| UZT-5 | Asia/Tashkent |
| GET-4 | Asia/Tbilisi |
| IRST-3:30IRDT,80/0,264/0 | Asia/Tehran |
| BTT-6 | Asia/Thimphu |
| JST-9 | Asia/Tokyo |
| ULAT-8 | Asia/Ulaanbaatar |

| | |
|---|---|
| CST-8 | Asia/Urumqi |
| ICT-7 | Asia/Vientiane |
| VLAT-10VLAST,M3.5.0,M10.5.0/3 | Asia/Vladivostok |
| YAKT-9YAKST,M3.5.0,M10.5.0/3 | Asia/Yakutsk |
| YEKT-5YEKST,M3.5.0,M10.5.0/3 | Asia/Yekaterinburg |
| AMT-4AMST,M3.5.0,M10.5.0/3 | Asia/Yerevan |
| AZOT1AZOST,M3.5.0/0,M10.5.0/1 | Atlantic/Azores |
| AST4ADT,M3.2.0,M11.1.0 | Atlantic/Bermuda |
| WET0WEST,M3.5.0/1,M10.5.0 | Atlantic/Canary |
| CVT1 | Atlantic/Cape Verde |
| WET0WEST,M3.5.0/1,M10.5.0 | Atlantic/Faroe |
| WET0WEST,M3.5.0/1,M10.5.0 | Atlantic/Madeira |
| GMT0 | Atlantic/Reykjavik |
| GST2 | Atlantic/South Georgia |
| GMT0 | Atlantic/St Helena |
| FKT4FKST,M9.1.0,M4.3.0 | Atlantic/Stanley |
| CST-9:30CST,M10.1.0,M4.1.0/3 | Australia/Adelaide |
| EST-10 | Australia/Brisbane |
| CST-9:30CST,M10.1.0,M4.1.0/3 | Australia/Broken Hill |
| EST-10EST,M10.1.0,M4.1.0/3 | Australia/Currie |
| CST-9:30 | Australia/Darwin |
| CWST-8:45 | Australia/Eucla |
| EST-10EST,M10.1.0,M4.1.0/3 | Australia/Hobart |
| EST-10 | Australia/Lindeman |
| LHST-10:30LHST-11,M10.1.0,M4.1.0 | Australia/Lord Howe |
| EST-10EST,M10.1.0,M4.1.0/3 | Australia/Melbourne |
| WST-8 | Australia/Perth |
| EST-10EST,M10.1.0,M4.1.0/3 | Australia/Sydney |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Amsterdam |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Andorra |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Athens |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Belgrade |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Berlin |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Bratislava |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Brussels |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Bucharest |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Budapest |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Chisinau |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Copenhagen |
| GMT0IST,M3.5.0/1,M10.5.0 | Europe/Dublin |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Gibraltar |
| GMT0BST,M3.5.0/1,M10.5.0 | Europe/Guernsey |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Helsinki |
| GMT0BST,M3.5.0/1,M10.5.0 | Europe/Isle of Man |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Istanbul |
| GMT0BST,M3.5.0/1,M10.5.0 | Europe/Jersey |

| | |
|---|---|
| EET-2EEST,M3.5.0,M10.5.0/3 | Europe/Kaliningrad |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Kiev |
| WET0WEST,M3.5.0/1,M10.5.0 | Europe/Lisbon |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Ljubljana |
| GMT0BST,M3.5.0/1,M10.5.0 | Europe/London |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Luxembourg |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Madrid |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Malta |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Mariehamn |
| EET-2EEST,M3.5.0,M10.5.0/3 | Europe/Minsk |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Monaco |
| MSK-4 | Europe/Moscow |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Oslo |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Paris |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Podgorica |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Prague |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Riga |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Rome |
| SAMT-3SAMST,M3.5.0,M10.5.0/3 | Europe/Samara |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/San Marino |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Sarajevo |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Simferopol |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Skopje |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Sofia |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Stockholm |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Tallinn |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Tirane |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Uzhgorod |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Vaduz |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Vatican |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Vienna |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Vilnius |
| VOLT-3VOLST,M3.5.0,M10.5.0/3 | Europe/Volgograd |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Warsaw |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Zagreb |
| EET-2EEST,M3.5.0/3,M10.5.0/4 | Europe/Zaporozhye |
| CET-1CEST,M3.5.0,M10.5.0/3 | Europe/Zurich |
| EAT-3 | Indian/Antananarivo |
| IOT-6 | Indian/Chagos |
| CXT-7 | Indian/Christmas |
| CCT-6:30 | Indian/Cocos |
| EAT-3 | Indian/Comoro |
| TFT-5 | Indian/Kerguelen |
| SCT-4 | Indian/Mahe |
| MVT-5 | Indian/Maldives |
| MUT-4 | Indian/Mauritius |

| | |
|---|---|
| EAT-3 | Indian/Mayotte |
| RET-4 | Indian/Reunion |
| WST11 | Pacific/Apia |
| NZST-12NZDT,M9.5.0,M4.1.0/3 | Pacific/Auckland |
| CHAST-12:45CHADT,M9.5.0/2:45,M4.1.0/3:45 | Pacific/Chatham |
| VUT-11 | Pacific/Efate |
| PHOT-13 | Pacific/Enderbury |
| TKT10 | Pacific/Fakaofo |
| FJT-12 | Pacific/Fiji |
| TVT-12 | Pacific/Funafuti |
| GALT6 | Pacific/Galapagos |
| GAMT9 | Pacific/Gambier |
| SBT-11 | Pacific/Guadalcanal |
| ChST-10 | Pacific/Guam |
| HST10 | Pacific/Honolulu |
| HST10 | Pacific/Johnston |
| LINT-14 | Pacific/Kiritimati |
| KOST-11 | Pacific/Kosrae |
| MHT-12 | Pacific/Kwajalein |
| MHT-12 | Pacific/Majuro |
| MART9:30 | Pacific/Marquesas |
| SST11 | Pacific/Midway |
| NRT-12 | Pacific/Nauru |
| NUT11 | Pacific/Niue |
| NFT-11:30 | Pacific/Norfolk |
| NCT-11 | Pacific/Noumea |
| SST11 | Pacific/Pago Pago |
| PWT-9 | Pacific/Palau |
| PST8 | Pacific/Pitcairn |
| PONT-11 | Pacific/Ponape |
| PGT-10 | Pacific/Port Moresby |
| CKT10 | Pacific/Rarotonga |
| ChST-10 | Pacific/Saipan |
| TAHT10 | Pacific/Tahiti |
| GILT-12 | Pacific/Tarawa |
| TOT-13 | Pacific/Tongatapu |
| TRUT-10 | Pacific/Truk |
| WAKT-12 | Pacific/Wake |
| WFT-12 | Pacific/Wallis |

## SetSNMP

Set the SNMP settings in the database.

**Parameters:**

---

***power_trap_en*** Enable/Disable power trap.

***input_loss_trap_en*** Enable/Disable input loss trap.

***input_recover_trap_en*** Enable/Disable input recover trap.

***temp_high_trap_en*** Enable/Disable temp high trap.

***temp_recover_trap_en*** Enable/Disable temperature recover trap.

***trap_hosts*** Trap hosts ipaddress. Default ip address value is 192.168.0.6.

***high_temp_suppress*** hiigh temperature. Default value is 30.

***nominal_temp_suppress*** nominal temperature default value is 30.

***snmp_v3_en*** Enable/Disable SNMP

***snmp_v3_encrypt*** SNMP v3 encrypt value.

***snmp_v3_user*** z3user.

***snmp_v3_passwd*** z3password.

***snmp_v3_usrpro*** snmp v3 usrpro default value is md5.

***snmp_v3_encrypt_passwd*** z3password.

***snmp_v3_encpro*** decrypt value.

***snmp_v3_engineid*** Default value for engineid is 0.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetSNMP response: {"ret":"0","status":"OK"}
```

---

## SetNMEAEnable

---

Save the GPS NMEA enable/disable value in the Database.

**Parameters:**

---

***nmea_enable*** Enable/Disable the NMEA. Possible value: on, off.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetNMEAEnable response: {"ret":"0","status":"OK"}
```

## SetOnvif

Save the ONVIF settings in the Database.

**Parameters:**

***fixed_profile_max*** Get the Maximum ONVIF profiles to allow (1 or 2).

***en_persistent_audio*** Enable/Disable ONVIF persistent audio channel.

***onvif_enable*** Enable/Disable ONVIF.

***list_all_video_sources*** List all the video source.

***ptz_timeout*** Get the PTZ timeout values.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetOnvif response: {"ret":"0","status":"OK"}
```

## OverlayStop

Stop overlay.

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***rgn_idx*** Index of overly for this channel.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

```
python3 z3overlay.py -ip 192.168.10.127 -username admin -passwd admin -aim -track -moon -rtsp
```

**Another method to execute:**

```
curl -s --connect-timeout 30 --max-time 60 --data "action=OverlayStop&chn=1&rgn_idx=1&source=welcome&type=text&location=top_center" "http://192.168.10.127/cgi-bin/control.cgi"
```

**Example output:**

```
{"ret":"0","status":" OK"}
```

## PtzAbsoluteMove

Send absolute move command to the PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzAuxiliary

---

Send PTZ auxiliary value to the PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzClearPreset

---

Remove the Ptz preset values for the user.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
action=PtzClearPreset&data={"idx":0,"profile_token":"profile1","preset_token":"2"}
```

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd admin
```

---

**Example output:**

```
PtzClearPreset response: {"ret":"0","status":"OK"}
```

---

## PtzContinuousMove

---

Send continous move command to the PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

Please refer the test code and steps to execute is available in this section with different speed and camera selection for both continuous and step mode

Pan to the left at half speed

```
action=PtzContinuousMove&data={"PanTilt":{"x":-0.5,"y":0},"idx":0}
```

Pan to the right at half speed

```
action=PtzContinuousMove&data={"PanTilt":{"x":0.5,"y":0},"idx":0}
```

Tilt down at max speed

```
action=PtzContinuousMove&data={"PanTilt":{"x":0,"y":1},"idx":0}
```

## PtzGotoPreset

---

Send preset value to PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action: "PtzGotoPreset", data: '{"idx":0,"preset_token":"2"}'
```

---

**Example output:**

```
PtzGotoPreset response: {"ret":"0","status":"OK"}
```

---

## PtzNewTourSpot

---

Get the new tour spot and update in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action=PtzNewTourSpot&data={"idx":0,"profile_token":"profile1","tour_token":
"1","name":"New_TourSpot"}
```

---

**Example output:**

```
PtzNewTourSpot response: {"ret":"0","status":"OK"}
```

---

## PtzPosition

---

Move the Ptz to the input position

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzPreset

---

Preset the Ptz based on the value.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzQuery

---

Get the ptz status and degree value from the ptz

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzRemoveTourSpot

---

Remove the ptz tour spot

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the profile token, preset tour token, row index.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action: "PtzRemoveTourSpot", data:
'{"profile_token":"profile1","tour_token":"1","row_index":0}'
```

---

**Example output:**

```
PtzRemoveTourSpot response: {"ret":0,"status":"OK"}
```

---

## PtzSetPreset

---

Set the preseting value in the PTZ

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action: "PtzSetPreset", data:
'{"idx":0,"profile_token":"profile1","name":"New_Preset"}'
```

---

**Example output:**

```
PtzSetPreset response: {"ret":"0","status":"OK","token":"2"}
```

---

## PtzStartTour

---

Start the PTZ tour

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
```

```
admin
```

```
action: "PtzStartTour", data:
'{"profile_token":"profile1","tour_token":"1","name":"New_TourSpot","idx":0}
'
```

---

**Example output:**

```
PtzStartTour response: {"ret":"0","status":"OK"}
```

---

## PtzStop

---

Stop the PTZ

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

```
action=PtzStop&data={"mode":1,"idx":0}
```

---

**Example output:**

```
PtzStop response: {"ret":"0","status":"OK"}
```

---

## RemoveJobs

---

Delete the schedule jobs from the Database.

**Parameters:**

---

***purgelist*** List of schedule jobs to be removed.

***rowcnt*** Number of rows.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## RestartBoard

---

Read the operation mode. Whether its encoding / decoding. Stop the channel encoding and restart the board

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
RestartBoard response: {"ret":"0","status":"OK"}
```

---

## SaveCamera

---

Saving the camera settings in the Database.

**Parameters:**

---

***zoom_direct_value*** Zoom value. Possible values: 0 (wide) to 0x7ac0 (full zoom)

***white_balance_mode*** White balance mode. Default value is auto. Possible values: auto / manual / indoor/ one-push / auto trace / sodium lamp auto / sodium lamp fixed / sodium lamp outdoor.

***color_gain*** color gain. Default value is 4. Possible values: Range from 0 (60%) to 14 (200%)

***color_hue*** color hue. Default value is 7. Possible values: Range from 0 (60%) to 14 (200%)

***chroma_suppress*** Chroma suppress. Default value is 1. Possible values: 0=none, 1, 2, 3

***wb_manual_rgain_direct*** white balance manual red gain. Default value is 190. Possible values: 0 to 255

***wb_manual_bgain_direct*** white balance manual blue gain. Default value is 190. Possible values: 0 to 255

***optical_zoom_only*** optical zoom only. Default value is 0. Possible values: 0, 1

***focus_direct_value*** focus direct value. Default value is 4096. Possible values: 0x1000 to 0xf000

***manual_focus*** Manual focus. Default value is auto. Possible values: auto, manual

***exposure_mode*** Exposure mode. Default value is 0. Possible values: 0 - Auto, 10 - Shutter Priority, 11 - Iris -Priority, 3 - Manual.

***shutter*** Shutter. Default value is 7.

***iris*** Iris. Default value is 0.

***gain*** Gain. Default value is 0.

***high_sensitivity*** high sensitivity value. Default value is 3. Possible values 0, 1.

***hlc_level*** hlc level. Default value is 0. Possible values in range 0 to 3.

**hlc_level_mask** mask value of hlc level. Default value is 0. Possible values 0, 1.

***stable_zoom*** stable zoom on or off. Default value is off.

***eflip*** Default value is 5. Possible values 0, 1.

***lr_reverse*** Default value is 0. Possible values 0, 1.

***monitor_mode*** Default value is 1080p-59.94.

***genlock_source*** Default value is 2.

***zoom_step_size*** Default value is 4. Possible values in range (1 - 7).

***focus_step_size*** Default value is 1. Possible values in range (0 - 7).

***manual_icr*** Default value is Off. Possible values: On, Off, Auto, Auto Color

***img_freeze*** Default value is 0. Possible values: 0, 1.

***hr_mode*** High resolution mode. Default value is 0. Possible values: 0, 1.

***img_stabilizer*** Default value is 0. Possible values 0, 1.

***img_bw*** Image black white. Default value is 0. Possible values 0, 1.

***nr_2d_level*** Default value is 5. Possible values in range (0 - 5).

***nr_3d_level*** Default value is 4. Possible values in range (0 - 5).

***icr_threshold*** Default value is 14. Possible values in range (0-255 for 4K camera, 0-28 for HD camera).

***slow_shutter*** Default value is 0. Possible values: 0, 1.

***slow_shutter_limit*** Default value is 4. Possible values in range: (1 to 6)

***flicker_reduction*** Default value is 0. Possible values: 0, 1.

***img_stabilizer_level*** Default value is 0. Possible values: 0, 2.

***wide_dynamic_range*** Default value is 3.

***ve_brightness*** Default value is 3. Possible values in range: 0, 6.

***ve_compensation_type*** Default value is 2. Possible values in range: 0, 3.

***ve_compensation_level*** Default value is 1. Possible values in range: 0, 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Save Camera response: {"ret":"0","status":"OK"}
```

---

## SaveCameraBoson

---

Save color pallete settings of boson camera in the Database.

**Parameters:**

---

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Save Camera Boson response: {"ret":"0","status":"OK"}
```

---

## SaveCameraDRSTamarisk

---

Save color pallete settings of DRS Tamarisk camera in the Database.

**Parameters:**

---

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Save Camera DRS Tamarisk response: {"ret":"0","status":"OK"}
```

## SaveCameraLink

Saving camera link settings in the Database.

**Parameters:**

***pxl_format*** Pixel format. Default value is 1.

***color_table*** Color table. Default value is whitehot.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Save Camera Link response: {"ret":"0","status":"OK"}
```

## SaveCameraLx

---

Saving camera Lx settings in the Database.

**Parameters:**

---

***SONY_ExposureMode*** Exposure Mode. Default value is 32852. Possible values: 32848 (Program Auto), 32849 (Aperture Priority), 32850 (Shutter Priority), 32851 (Manual Exposure), 32852 (Intelligent Auto).

***name*** Name of the camera. Default value is s3ca_1.

***SONY_ShutterSpeed*** Shutter speed. Defualt value is NULL.

***SONY_ISO*** ISO. Defualt value is NULL.

***SONY_ExposureComp*** Exposure comp. Defualt value is NULL.

***SONY_WhiteBalance*** White balance. Defualt value is NULL.

***SONY_ColorTemp*** Color temperature. Defualt value is NULL.

***SONY_APS_C*** APS_C. Defualt value is NULL.

***SONY_NtscPalSelect*** NTSC / PAL selection. Defualt value is NULL.

***SONY_MovieSteadyMode*** Movie steady mode. Defualt value is NULL.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Save Camera Lx response: {"ret":"0","status":"OK"}
```

## SaveCameraTau

Saving camera TAU settings in the Database.

**Parameters:**

***color_table*** color table value .

***is_active*** Active/Deactive. Default value is 0.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SaveJobs

Save the schedule jobs in the Database.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the schedule activities.**.**

***rowcnt*** Get the number of row count.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SavePresets

---

Check the operation mode whether its in encoder or decoder. Save the preseting value of the mode in the Database.

**Parameters:**

---

***dec_current_preset*** Save the decoder current preseting values.

***enc_current_preset*** Save the encoder current preseting values.

***preset*** Settings for encoder or decoder.

***enc_channels*** Number of channels. Maximum number of channels is 2 for decoder and 4 for encoder.

***opmode*** Operation mode. Default is Encoder. Possible values: Encoder / Decoder.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SavePresets response: {"ret":"0","status":"OK"}
```

## SaveUser

Save the state, preset, encoder/decoder settings in the Database.

**Parameters:**

***enc_current_preset*** Get the encoder preseting values for the current channel.

***dec_current_preset*** Get the decoder preseting values for the current channel.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SaveUser response: {"ret":"0","status":"OK"}
```

## SetAdv

Save the encoder advance settings in the Database.

**Parameters:**

***enc_adv_setting*** Get the encoder advance setting values.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetAdv response: {"ret":"0","status":"OK"}
```

## SetAPConfig

Set the wifi and its password.

**Parameters:**

***passwd*** Get the wifi password

***passwden*** Get wifi enable / disable. Default value is disable.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetAPConfig response: {"ret":"0","status":"OK"}
```

---

## SetAudio

---

Save the audio input settings in the Database.

**Parameters:**

---

***aport*** Get the audio port. Default value is MIC. Possible values are MIC , MICL

***analog_gain_db*** Get the analog gain decibal value. Default value is 0. The value ranges are -97 to 30.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetAudio response: {"ret":"0","status":"OK"}
```

---

## SetCameraLink

---

Save the color table and pixel format values in the Database and set the color table and pixel format in the fpga.

**Parameters:**

---

***val*** Get the value for color table and pixel format.

***opt*** The options are color table and pixel format.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Camera Link response: {"ret":"0","status":"OK"}
```

---

## SetConsole

---

Set the console value as ttyAMA0 or none.

**Parameters:**

---

***console_enable*** Get console value is enabled / disabled.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Console response: {"ret":"0","status":"OK"}
```

---

## SetDDNSEnable

Set the Dynamic Domain Name System settings in the Database.

**Parameters:**

***ddns_enable*** Get the ddns enable value. Default value is Off. Possible values; (On, Off)

***ddns_provider*** Get the ddns provider. Default value is freedns. Possible values: *(*freedns, freemyip, dyn, domains.google.com, duckdns.org, no-ip.com, tunnelbroker.net, dynv6.com, cloudxnv.net, dnspod.cn, cloudflare.com).

***ddns_username*** Provide the username.

***ddns_password*** Provide the password.

***ddns_hostname*** Hostname for DDNS login.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
Set DDNS Enable response: {"ret":"0","status":"OK"}
```

## SetDebugLevel

---

Set the logging level. The setting is persistent.

**Parameters:**

---

***sysdebuglevel*** Get the system debug level. Possible values 0=None, 1=Error, 2=Information, 3=Diagnostics, 4=Codeflow, 5=Dataflow.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Debug Level response: {"ret":"0","status":"OK"}
```

---

## SetDevName

---

Set the device name in the Database.

**Parameters:**

---

***sysdevicename*** Get the system device name.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Dev Name response: {"ret":"0","status":"OK"}
```

---

## SetDisplay

---

Controls composite output (passthru video from camera).

**Parameters:**

---

***disp_std*** Get display standard for encoder. Default value is Auto.

***disp_input*** Get display input for encoder. Default value is MACRO_DEFAULT_DISP_INPUT.

***save_only*** Possible values: (True / False)

***disp_layout*** Display layout for decoder. Default value is 1×1.

***disp_mode*** Display mode for decoder. Default value is UltraHD.

***disp_std2*** Display standard for decoder. Default value is auto.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
Set Display response: {"ret":"0","status":"OK"}
```

---

## SetDSCP

---

Set the DSCP configuration settings in the Database.

**Parameters:**

---

***diff_serve*** Get the DSCP configuration value. Default value is 0x00. Possible values are ( 0x00, 0xB8, 0x88, 0x90, 0x98, 0x80, 0x68, 0x70, 0x78, 0x60, 0x48, 0x50, 0x58, 0x28, 0x30, 0x38, 0x01).

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetDSCP response: {"ret":"0","status":"OK"}
```

## SetFpga

Set the fpga settings in the Database.

**Parameters:**

***fpgafileglob*** Get fpga value. Possible values are (boson / cvbs / tamarisk / tau2).

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetFpga response: {"ret":"0","status":"OK"}
```

---

## SetIp

---

Control networking settings of the Z3 device such as local IPv4 IP address, netmask, etc. Additionally control Encoder Auto-Start after boot-up and enable/disable showing of advanced WebUI settings.

**Parameters:**

---

***do_autostart*** Control automatic stream start after bootup. Possible values: 1 = do autostart 0 = do not autostart.

***enc_adv_setting*** Control appearance of advanced settings on WebUI. Possible values: off, on.

***ipmtu*** Control Ethernet IP Maximum Transmission Unit (MTU) size in bytes. Possible values: 576 to 1500.

***eth_speed*** Control Gigabit Ethernet auto-negotiation allowed speeds. Possible values: AUTO, AUTO-100, 1000, 100, 10.

***eth_duplex*** Control Ethernet auto-negotiation allowed duplex. Possible values: AUTO, FULL, HALF.

***local_ip*** IPv4 address.

***local_netmask*** IPv4 netmask.

***default_gw*** IPv4 default gateway.

***local_dnsip*** DNS server primary.

***local_dnsip2*** DNS server secondary.

***use_dhcp*** Controls use of DHCP or static IP. Possible values: 1 (DHCP enabled), 0 (DHCP disabled, use static IP).

***local_hostname*** Network hostname.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=SetIp&default_gw=172.29.0.1&do_autostart=1&enc_adv_setting=onð_duplex
=AUTOð_speed=AUTO&ipmtu=1500&local_dnsip=8.8.8.8&local_dnsip2=4.4.4.4&local_
hostname=Z3Dome-4M&local_ip=172.29.10.122&local_netmask:
=255.255.0.0&use_dhcp=1
```

---

**Example output:**

```
SetIp response: {"ret":"0","status":"OK"}
```

---

## SetLoginLimit

---

Set the login limit settings in the Database

**Parameters:**

---

***maxlogin*** mximum number of login limit. Possible value in range (3 to 99 ).

***login_window*** Login window period, the value are considered in seconds. Possible value in range (60 to 32768).

***lockout_interval*** Lockout interval values are mentioned in seconds, which is used when maximum number of login limit is reached. Possible value in range (-1 to 32768)

***login_timeout*** Login timeout

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetLoginLimit response: {"ret":"0","status":"OK"}
```

---

## SetNFS

---

Set the NFS settings in the Database.

**Parameters:**

---

***nfs_enable*** Enable/Disable nfs

***nfs_server*** Get the nfs server ip address. Default is 192.168.1.6

***nfs_server_root*** Get the server root path / location.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetNFS response: {"ret":"0","status":"OK"}
```

---

## SetSNTP

---

Set SNTP (Simple Network Time Protocol) settings in the Database.

**Parameters:**

---

***enable*** Enable/Disable SNTP. Possible value: true, false. Default value true.

***servers*** NTP server or list of NTP servers.

***timezone*** Linux TZ database value for Timezone.

***timezone_name*** Name for a time zone. Default is America/Chicago.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetSNTP response: {"ret":"0","status":"OK"}
```

## SetSNMP

Set the SNMP settings in the database.

**Parameters:**

***power_trap_en*** Enable/Disable power trap.

***input_loss_trap_en*** Enable/Disable input loss trap.

***input_recover_trap_en*** Enable/Disable input recover trap.

***temp_high_trap_en*** Enable/Disable temp high trap.

***temp_recover_trap_en*** Enable/Disable temperature recover trap.

***trap_hosts*** Trap hosts ipaddress. Default ip address value is 192.168.0.6.

***high_temp_suppress*** hiigh temperature. Default value is 30.

***nominal_temp_suppress*** nominal temperature default value is 30.

***snmp_v3_en*** Enable/Disable SNMP

***snmp_v3_encrypt*** SNMP v3 encrypt value.

***snmp_v3_user*** z3user.

***snmp_v3_passwd*** z3password.

***snmp_v3_usrpro*** snmp v3 usrpro default value is md5.

***snmp_v3_encrypt_passwd*** z3password.

***snmp_v3_encpro*** decrypt value.

***snmp_v3_engineid*** Default value for engineid is 0.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetSNMP response: {"ret":"0","status":"OK"}
```

---

## SetNMEAEnable

---

Save the GPS NMEA enable/disable value in the Database.

**Parameters:**

---

***nmea_enable*** Enable/Disable the NMEA. Possible value: on, off.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetNMEAEnable response: {"ret":"0","status":"OK"}
```

---

## SetOnvif

---

Save the ONVIF settings in the Database.

**Parameters:**

---

***fixed_profile_max*** Get the Maximum ONVIF profiles to allow (1 or 2).

***en_persistent_audio*** Enable/Disable ONVIF persistent audio channel.

***onvif_enable*** Enable/Disable ONVIF.

***list_all_video_sources*** List all the video source.

***ptz_timeout*** Get the PTZ timeout values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetOnvif response: {"ret":"0","status":"OK"}
```

## SetOnvifVMD

Save the ONVIF video motion detection and too darkness settings in the Database.

**Parameters:**

***vmd_enable*** Enable/Disable video motion detection.

***vmd_sens*** sensitivity value required for vmd. Possible value in range: (1 to 100).

***vmd_zone_modify*** Enable/Disable video motion and too darkness zone coordinates.

***vmd_vcrop_width*** crop width of the coordinates.

***vmd_vcrop_height*** crop height for VMD and too_darkness.

***vmd_vcrop_x*** crop startx for VMD and too_darkness.

***vmd_vcrop_y*** crop starty for VMD and too_darkness.

***td_enable*** Enable/Disable too darkness.

***td_thresh*** Threshold value for too darkness. Possible values in range: 1 to 100.

***vmd_td_channel*** Get the channel number. Possible values: 1, 2.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetOnvifVMD response: {"ret":"0","status":"OK"}
```

---

## SetRTSP

---

Set the RTSP settings in the Database.

**Parameters:**

---

***rtspd_port*** Get rtsp port. Default value is 554. Ports that are not in use are valid; ranging from 0 to 65535 .

***rtspd_timeout*** Timeout value required for the rtsp connection. Default is 60. The range is from -1 to 32767. It is not recommended to lower than 60.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetRTSP response: {"ret":"0","status":"OK"}
```

---

## SetTermSrvEnable

---

Sett the term server value in the Database.

**Parameters:**

---

***termserve_remote_enable*** Enable/Disable remote access to terminal server. Possible values are : on, off.

***termserve_shared_enable*** Enable/Disable shared access to terminal server. Possible values are : on, off.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetTermSrvEnable response: {"ret":"0","status":"OK"}
```

---

## SetVideoGroup

---

Set the video group configuration settings in the Database.

**Parameters:**

---

***group*** Video config groups. Default value 0.

***nr_enable*** Enable / Disable NR. Default value is on. Possible value: on/off.

***crop_enable*** Enabled / Disable the crop.

***crop_x*** Crop x coordinates.

***crop_y*** Crop y coordinates.

***crop_width*** Crop width

***crop_height*** Crop height

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetVideoGroup response: {"ret":"0","status":"OK"}
```

---

## SetViewport

Set the view port for the decoder and update in the Database.

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***viewport*** viewport settings. The default value is fit.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetViewport response: {"ret":"0","status":"OK"}
```

## SetViVpssMode

Set the ViVpss mode settings in the Database.

**Parameters:**

---

***enc_vivpss_mode_enable*** Enable/Disable the vivpss mode. Default value is off. Possible values: on, off.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetViVpssMode response: {"ret":"0","status":"OK"}
```

---

## SetWifiAP

---

Set the wifi network settings in the Database.

**Parameters:**

---

***ssid*** Get the Wifi service set identifier value.

***psk*** Get the Wifi password.

***wifi_client_local_ip*** Local IP addres.

***wifi_client_local_local_netmask*** Subnet mask for Wifi network.

***wifi_client_default_gw*** Default Gateway be used for Wifi Network.

***wifi_client_use_dhcp*** DHCP server used with Wifi network for DNS resolutions.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetZFinderEnable

---

Save ZFinder enable / disable in the Databse.

**Parameters:**

---

***zfinder_enable*** Get ZFinder enable/disable value. Possible values: off, on

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

### Command to execute:

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetZFinderEnable response: {"ret":"0","status":"OK"}
```

---

## StartChannel

---

Start the encoder channel and update the state as running in the Database. Once you start the channel, it will not transmit data until the video input is detected.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***actv_preset***

***vsource*** video source. Default value is Camera1. Possible values: Camera1 / Camer2 / HDMI1.

***vres*** video resolution. Default value is Follow Input. Possible values: Follow Input, 3840 x 2160, 1920 x 1080, 1440 x 1080, 1280 x 1024, 1280 x 720, 1024 x 768, 960 x 720, 1024 x 576.

***vcodec*** video codec. Default value is H265. Possible values: H265, H264, MJPEG, none.

***vgdr*** video gradual data refresh. Default value is on. Possible value on, off.

***vprofile*** video profile. Default value is high. Possible value high, main, baseline.

***vratectrl*** video rate control. Default value is cbr. Possible value cbr, vbr.

**vbitrate** video bit rate.

***vframeratediv*** video frame rate. Default value is 1. Possible values: 1 (Full), 2 (Half), 4 (Quarter), 6 (Sixth).

***vgopsize*** video group of picture size. Default value is 60. Possible values: 1 (1 Frame Only), 5 (5 Frames), 8 (8 Frames), 10 (10 Frames), 12 (12 Frames), 15 (15 Frames), 25 (25 Frames), 30 (30 Frames), 50 (50 Frames), 60 (60 Frames), 90 (90 Frames), 100 (100 Frames), 120 (120 Frames), 200 (200 Frames), 240 (240 Frames).

***vprotocol*** video protocol.

***vdest*** video destination port.

***vpid*** video packet identifier. Default value is 221.

***vdelay*** video delay. Default value is 300.

***pcrpid*** PCR packet identifier. Default value is 521.

***pcrinterval*** Default value is 50.

***pmtpid*** program map table packet identifier. Default value is 31.

***tsrate*** bit rate for transport stream. Default value is 5000K.

***tslowlat*** transport stream low latency mode. Default value is off. Possible values: on, off.

***feconoff*** Forward error correction in transport stream enable/ disable. Default value is off. Possible values: on, off.

***fecrow*** Forward error correction row. Default value is 1.

***feccol*** Forward error correction column, Default value is 5.

***zixioverhead*** ZIXI Forward Error Correction total overhead as a percentage of transport rate. Default value is 15.

***zixilatency*** ZIXI target latency (to allow ARQ) in milliseconds. Default value is 500.

***zixifecblock*** ZIXI Forward Error Correction block size in milliseconds. Max value is 1/2 of Zixi Latency. Default value is 50.

***zixirateadjen*** ZIXI dynamic bitrate adjustment according to network conditions. Default value is on. Possible values: on, off.

***zixiauthen*** ZIXI dynamic authentication enable/disable . Default value is off. Possible values: on, off.

***zixisession*** ZIXI session id. Default value is test.

***zixiuser*** ZIXI user identifier. Default value is user.

***aenable*** Audio enable/disable. Default value is on. Possible values: on, off.asource. Default value is MICL.

***apair*** Stereo pair select for digital inputs. Default value is 0. Possible values: (0 (1 2), 1 (3 4), 2 (5 6), 3 (7 8), 260 (1 2,3 4), 548 (1 2,3 4,5 6), 1252 (1 2,3 4,5 6,7 8)).

***acodec*** Audio codec. Default value is fdk_aaclc.

***abitrate*** Audio bit rate. Default value is 128000.

***asamplerate*** Audio sample rate. Default value is 48000.

***aport*** Audio port. Default value is 8700.

***apid*** Audio packet identifier. Default value is 120.

***aptspcr*** Maximum difference between the PTS and PCR in the audio stream Default value is 250.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
StartChannel response: {"ret":"0","status":"RUNNING"}
```

---

## StartMTS

---

Start the MTS.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
StartMTS response: {"ret":"0","status":"OK"}
```

---

## StopChannel

---

Stop the encoder channel and update the state in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
StopChannel response: {"ret":"0","status":"STOPPED"}
```

## StopMTS

Stop the MTS.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
StopMTSresponse: {"ret":"0","status":"OK"}
```

## TempStatus

Get the current temperature status for CPU and lense.

**Parameters:**

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
TempStatus response: {"ret":"0","status":"OK"}
```

---

## update_require_web_login

---

Update/Change whether web login credential required or not in the Database.

**Parameters:**

---

***require*** Required web login value. Possible values: 1 or 0

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
update_require_web_login response: {"ret":"0","status":"OK"}
```

---

## UpdatePtz

---

Update the PTZ value in the Database.

#### Parameters:

---

***data*** Parse the JSON formatted data and retrieve the PTZ values.

#### Returns:

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=UpdatePtz&rowcount=1&data={"row0":{"db_videosource":"CAMERA","token":
"ptz","name":"videoin1","enabled":1,"pelcoaddr":1,"tcpport":2000,"ptztype":"
relative","flip":"none","pos":"simulated","maxtilt":90,"rowid":1}}
```

---

**Example output:**

```
UpdatePtz response: {"ret":"0","status":"OK"}
```

---

## UpdatePtzPreset

---

Update the PTZ preset value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ preset values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=UpdatePtzPreset&rowcount=2&data={"row0":{"profiletoken":"profile1","t
oken":"1","name":"home","action":null,"rowid":1},"row1":{"profiletoken":"pro
file1","token":"2","name":"temp","action":null,"rowid":2}}
```

---

**Example output:**

```
UpdatePtzPreset response: {"ret":"0","status":"OK"}
```

---

## UpdatePtzTourSpots

---

Update the PTZ tour spots value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ tour spots values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=UpdatePtzTourSpots&rowcount=1&data={"row0":{"ProfileToken":"profile1"
,"PresetTourToken":"1","PresetDetailToken":"1","StayTime":5,"RowIndex":0,"ac
tion":"blank","rowid":1}}
```

---

**Example output:**

```
UpdatePtzTourSpots response: {"ret":"0","status":"OK"}
```

---

## UpdateTerm

---

Control serial port terminal server for VISCA

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the terminal server values.

***term_mode*** Possible values: client, server

***term_protocol*** Possible values: clearchannel, telnet

***term_localport*** Local TCP port (for server mode)

***term_servaddr*** Remote IP address (for client mode only)

***term_servport*** Remote TCP port (for client mode only)

***term_data_bits*** number of data bits.

***term_parity*** Possible values: None, Odd, Even, Mark, Space

***term_stop_bits*** Stop bits sent at the end of every character.

***term_baudrate*** Possible values: 9600, 19200, 38400, 57600, 115200

***term_devicefile*** device fine name

***term_function*** functionality

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=UpdateTerm&rowcount=5&data={"row0":{"term_devicefile":"/dev/ttyHS1","
term_baudrate":"9600","term_data_bits":"8","term_parity":"None","term_stop_b
its":"1","term_function":"rs485","term_localport":"2000","rowid":1},"row1":{
"term_devicefile":"/dev/ttyHS3","term_baudrate":"9600","term_data_bits":"8",
"term_parity":"None","term_stop_bits":"1","term_function":"camera1_visca","t
erm_localport":"1000","rowid":2},"row2":{"term_devicefile":"/dev/ttyHS2","te
rm_baudrate":"9600","term_data_bits":"8","term_parity":"None","term_stop_bit
```

```
s":"1","term_function":"ir_led_zoom_ctrl","term_localport":"1001","rowid":3}
,"row3":{"term_devicefile":"/dev/ttyMSM0","term_baudrate":"115200","term_dat
a_bits":"8","term_parity":"None","term_stop_bits":"1","term_function":"conso
le","term_localport":"1800","rowid":4},"row4":{"term_devicefile":"/dev/ttyHS
4","term_baudrate":"2400","term_data_bits":"8","term_parity":"None","term_st
op_bits":"1","term_function":"user","term_localport":"1600","rowid":5}}
```


---

**Example output:**

```
UpdateTerm response: {"ret":"0","status":"OK"}
```

---

## user_add

---

Add a new user in the Database.

**Parameters:**

---

***Name*** Get new user name

***Level*** Get the level of permission for the new user name. Possible value: 0 to 7.

***Password*** Get the password for the new user name.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
user_add response: {"ret":"0","status":"OK"}
```

## user_remove

Remove the user from the database.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the user name.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action: "user_remove", data: '{"Name":"temp"}'
```

**Example output:**

```
user_remove response: {"ret":"0","status":"OK"}
```

## user_update_level

---

Update/change the level of permission to the given user name.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the user name and level of the permission.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
user_update_level response: {"ret":"0","status":"OK"}
```

---

## user_update_password

---

Update/change the password to the given user name.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the user name and password.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
user_update_password response: {"ret":"0","status":"OK"}
```

---

## download_remove

---

```
http://Encoder_IP/cgi-bin/control.cgi?action=download_remove&data=FILENAME

http://192.168.0.120/cgi-bin/control.cgi?action=download_remove&data=/media/
sda/MOV1_000040.mp4
```

remove the downloaded media content from the usb drive or sd card.

**Parameters:**

---

***data*** Filename from the downloaded media content to be removed from the removable media device.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

# Generic Control (GET)

## 8021x

---

Get the exthernet 802x information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***interface*** TEXT,

***enable*** TEXT.

***eap*** TEXT.

***anonymous_identity*** TEXT.

***identity*** TEXT.

***password*** TEXT.

***private_key_password*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
8021x response:
{"anonymous_identity":"","ca_cert_status":"Unavailable","eap":"peap","enable
":"off","identity":"","interface":"eth0","password":"","priv_cert_status":"U
navailable","priv_key_status":"Unavailable","private_key_password":"","ret":
"0"}
```

---

## GetCronJobs

---

Get the schedule job from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***enable*** INTEGER,

***cron*** TEXT.

***func*** TEXT.

***func_args*** TEXT.

***type*** TEXT.

***minutes*** INTEGER.

***hours*** INTEGER.

***day*** INTEGER.

***month*** INTEGER.

***cron*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
```

```
admin
```

---

**Example output:**

```
GetCronJobs response:
{"data":[],"metadata":[{"datatype":"integer","editable":true,"label":"Enable
","name":"enable"},{"datatype":"text","editable":true,"label":"Name","name":
"name"},{"datatype":"text","editable":true,"label":"Function","name":"func"}
,{"datatype":"text","editable":true,"label":"Function
Arguements","name":"func_args"},{"datatype":"text","editable":true,"label":"
Type","name":"type"},{"datatype":"integer","editable":true,"label":"Minutes"
,"name":"minutes"},{"datatype":"integer","editable":true,"label":"Hours","na
me":"hours"},{"datatype":"integer","editable":true,"label":"Day","name":"day
"},{"datatype":"integer","editable":true,"label":"Month","name":"month"},{"d
atatype":"text","editable":true,"label":"Cron
Command","name":"cron"}],"ret":"0"}
```

---

## GetNFSMount

---

Get the Network File Sharing mounted infromation from the device.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mounted*** TEXT,

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Get NFS Mount response: {"mounted":256,"ret":"0"}
```

## aspect_info

Get the video resolution for the current channel from the Database and calculate the aspect ratio.

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***numerator*** INTEGER,

***denominator*** INTEGER.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Aspect Info response:
{"denominator":9,"numerator":16,"ret":"0","status":"OK"}
```

## boardinfo

Get the board and model information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***logo_enable*** TEXT,

***logo_filename*** TEXT.

***logo_width*** INTEGER.

***logo_height*** INTEGER.

***logo_blob*** BLOB.

***ico_blob*** BLOB.

***model_enable*** TEXT.

***model_name*** TEXT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Boardinfo response:
{"MODEL":"FV4K-13A","board_id":"0xFF21","hw50serial":"50014401222610007","hw
serial":"30014410221610012","hwversion":"FV4K-13A","logo_enable":"off","logo
```

```
_height":156,"logo_width":319,"macaddr":"40:cd:3a:06:10:bc","model_enable":"
off","model_name":"Z3-
Encoder","opmode":"encoder","opstate":"RUNNING","processor_id":"cv22bub","re
t":"0","sensor_serial":"30014410221610012\n","sysdevicename":"Z3Cam","z3_avm
ux":"enabled","z3_klv":"enabled","z3_klv_serial_only":"enabled","z3_sntp":"e
nabled","z3_termsrv":"enabled","z3_tslowlat":"enabled","z3_webproxy":"enable
d"}
```


---

## cam_state

---

Get the current camera state information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT,

***enc_channels*** TEXT.

***opmode*** TEXT.

***enc_current_preset*** TEXT.

***dec_current_preset*** TEXT.

---

**Command to execute and Example :**

Please refer in this document in the below section cam_state (sys_DIGEST_ARGPARSE.py)

---

## camera

---

Get the current camera settings from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***zoom_direct_value*** INTEGER.

***white_balance_mode*** TEXT.

***color_gain*** INTEGER.

***color_hue*** INTEGER.

***chroma_suppress*** INTEGER.

***wb_manual_rgain_direct*** INTEGER.

***wb_manual_bgain_direct*** INTEGER.

***visca_localport*** TEXT.

***optical_zoom_only*** BOOL. 0 or 1.

***focus_direct_value*** INTEGER.

***manual_focus*** TEXT.

***flexio_localport*** TEXT.

***exposure_mode*** INTEGER.

***shutter*** INTEGER.

***iris*** INTEGER.

***gain*** INTEGER.

***high_sensitivity*** INTEGER.

***hlc_level*** INTEGER.

***hlc_level_mask*** INTEGER.

***stable_zoom*** INTEGER.

***eflip*** INTEGER.

***lr_reverse*** INTEGER.

***monitor_mode*** TEXT.

***genlock_source*** INTEGER.

***manual_icr*** TEXT.

***zoom_step_size*** INTEGER.

***focus_step_size*** INTEGER.

***img_freeze*** INTEGER.

***hr_mode*** INTEGER.

***img_stabilizer*** INTEGER.

***img_bw*** INTEGER.

***nr_2d_level*** INTEGER.

***nr_3d_level*** INTEGER.

***icr_threshold*** INTEGER.

***slow_shutter*** BOOL. 0 or 1.

***slow_shutter_limit*** INTEGER.

***flicker_reduction*** BOOL. 0 or 1.

***img_stabilizer_level*** INTEGER.

***wide_dynamic_range*** TEXT.

***ve_brightness*** TEXT.

***ve_compensation_type*** TEXT.

***ve_compensation_level*** TEXT.

***tab_index*** TEXT.

***imgvflip*** INTEGER.

***imghflip*** INTEGER.

***imgrotate*** INTEGER.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**


```
Camera response:
{"backlight":3,"chroma_suppress":1,"color_gain":4,"color_hue":7,"dbversion":
"1.1","eflip":3,"exposure_mode":0,"flexio_localport":"1500","flicker_reducti
on":0,"focus_direct_value":4096,"focus_step_size":1,"gain":4,"genlock_source
":2,"high_sensitivity":3,"hlc_level":0,"hlc_level_mask":0,"hr_mode":3,"icr_t
hreshold":-1,"img_bw":0,"img_freeze":3,"img_stabilizer":3,"img_stabilizer_le
vel":0,"imghflip":0,"imgrotate":0,"imgvflip":0,"iris":15,"lr_reverse":3,"man
ual_focus":"auto","manual_icr":"off","monitor_mode":"2160p-29.97","nr_2d_lev
el":5,"nr_3d_level":4,"optical_zoom_only":0,"ret":"0","shutter":7,"slow_shut
ter":0,"slow_shutter_limit":-1,"stable_zoom":0,"tab_index":"1","ve_brightnes
s":"-1","ve_compensation_level":"-1","ve_compensation_type":"-1","visca_loca
lport":"1000","wb_manual_bgain_direct":190,"wb_manual_rgain_direct":190,"whi
te_balance_mode":"auto","wide_dynamic_range":"3","zoom_direct_value":0,"zoom
_step_size":4}
```


---

## camera_exposure

---

Get the current camera exposure values from the visca camera.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***ae_mode*** TEXT,

***shutter*** TEXT.

***iris*** TEXT.

***gain_inquiry*** TEXT.

***sensitivity*** TEXT.

***hls_inquiry*** TEXT.

***min_shutter*** INTEGER.

***max_shutter*** INTEGER.

***shutter_label*** TEXT.

***min_iris*** INTEGER.

***max_iris*** INTEGER.

***iris_label*** TEXT.

***min_gain*** INTEGER.

***max_gain*** INTEGER.

***gain_label*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Camera Exposure response:
{"ae_mode":0,"backlight":3,"gain":10,"gain_label":["0db",null,null,null,null
,null,null,null,"24db",null,null,null,null,null,null,null,"48db"],"hlc":0,"i
ris":25,"iris_label":["0",null,null,null,null,null,null,null,null,null,null,
null,null,null,null,"F4.8",null,null,null,null,null,null,null,null,"F2.0"],"
max_gain":17,"max_iris":25,"max_shutter":33,"min_gain":1,"min_iris":0,"min_s
hutter":6,"ret":"0","sensitivity":3,"shutter":16,"shutter_label":["1/1",null
,null,null,null,null,null,null,null,null,null,null,null,null,"1/100",null,nu
ll,null,null,null,null,null,null,null,null,null,"1/10K"]}
```

---

## camera_monitor_mode

---

Get the current camera monitor mode values from the visca camera.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***modes*** TEXT.

***current_mode*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Camera Monitor Mode response:
{"current_mode":"2160p-29.97","modes":{"1080p-25":8,"1080p-29.97":6,"1080p-5
0":20,"1080p-59.94":19,"2160p-25":30,"2160p-29.97":29,"720p-50":12,"720p-59.
94":9},"ret":"0"}
```

---

## dec

---

Get the current decoder settings from the Database.

**Parameters:**

---

***chn*** Get the decoder channel number. The default decoder channel is 1. Select between decoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT.

***channel*** INTEGER.

***url*** TEXT.

***aenable*** TEXT.

***viewport*** TEXT.

***latency_mode*** TEXT.

***max_latency_ms*** TEXT.

***srt_mode*** TEXT.

***srt_latency*** TEXT.

***srt_decrypt*** TEXT.

***srt_pass*** TEXT.

***zixi_fec*** TEXT.

***zixi_fecoverhead*** TEXT.

***zixi_fecblock*** TEXT.

***zixi_latency*** TEXT.

***zixi_decrypt*** TEXT.

***zixi_pass*** TEXT.

***rtsp_flags*** TEXT.

---

## download

---

Get the downloaded file information from the removable storage media.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***job*** Optional***.*** The job from the schedule tab.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***FileName*** TEXT.

***FileSize*** TEXT.

***LastModified*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Download response:
{"data":[],"metadata":[{"datatype":"text","editable":false,"label":"File
Name","name":"download_filename"},{"datatype":"integer","editable":false,"la
bel":"File
Size","name":"download_size"},{"datatype":"text","editable":false,"label":"L
ast Modified","name":"download_lastmodified"}],"ret":"0"}
```

---

## drs_tamarisk_settings

Get the camera model of drs tamarisk settings fromt the Database.

**Parameters:**

***job*** Optional***.*** The job from the schedule tab.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***color_pallette*** TEXT.

***tab_index*** TEXT.

***color_pallette_disabled*** TEXT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
DRS Tamarisk Settings response:
{"color_pallette":"white_hot","color_pallette_disabled":"false","ret":"0","t
ab_index":"1"}
```

## enc

Get the current channel encoder settings from the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT,

***channel*** INTEGER.

***vbitrate*** TEXT.

***vframeratediv*** INTEGER.

***vsource*** TEXT.

***vcodec*** TEXT.

***vprofile*** TEXT.

***vprotocol*** TEXT.

***avmux_index*** TEXT.

***vgdr*** TEXT.

***vdest*** TEXT.

***aenable*** TEXT.

***acodec*** TEXT.

***abitrate*** TEXT.

***asamplerate*** TEXT.

***aport*** TEXT.

***apair*** INTEGER.

***vres*** TEXT.

***vgopsize*** INTEGER.

***vpid*** INTEGER.

***apid*** INTEGER.

***zixiauthen*** TEXT.

***zixisession*** TEXT.

***zixiuser*** TEXT.

***zixioverhead*** TEXT.

***zixifecblock*** TEXT.

***zixilatency*** TEXT.

***zixirateadjen*** TEXT.

***fecrow*** INTEGER.

***feccol*** INTEGER.

***feconoff*** TEXT.

***aptspcr*** INTEGER.

***tslowlat*** TEXT.

***tsrate*** TEXT.

***pcrpid*** INTEGER.

***pcrinterval*** INTEGER.

***pmtpid*** INTEGER.

***klvenable*** TEXT.

***klvmode*** TEXT.

***klvmuxmethod*** TEXT.

***klvsrc*** TEXT.

***klvbrate*** TEXT.

***klvserialbaud*** TEXT.

***klvpid*** TEXT.

***vratectrl*** TEXT.

***vdelay*** INTEGER.

***storage*** TEXT.

***fprefix*** TEXT.

***asource*** TEXT.

***authonoff*** TEXT.

***auth_user*** TEXT.

***auth_passwd*** TEXT.

***auxonoff*** TEXT.

***filesize*** TEXT.

***nfstrength*** TEXT.

***telopenable*** TEXT.

***teloptext*** TEXT.

***teloplocation*** TEXT.

***telopcharsize*** TEXT.

***teloptextcolor*** TEXT.

***telopoutlineenable*** TEXT.

***telopoutlinecolor*** TEXT.

***gps_overlay_enable*** TEXT.

***gps_overlay_device*** TEXT.

***gps_overlay_location*** TEXT.

***gps_overlay_char_size*** TEXT.

***pipenable*** TEXT.

***piplocation*** TEXT.

***vquality*** TEXT.

***vinterlacemode*** TEXT.

***vmulticastdest*** TEXT.

***amulticastdest*** TEXT.

***rtsp_auth_enable*** TEXT.

***rtsp_auth_username*** TEXT.

***rtsp_auth_password*** TEXT.

***rtsp_transport_mode*** TEXT.

***lowdelay_opt*** TEXT.

***vcropaspect*** TEXT.

***vcrop_enable*** TEXT.

***vcrop_width*** INTEGER.

***vcrop_height*** INTEGER.

***vcrop_x*** INTEGER.

***vcrop_y*** INTEGER.

***rotate_enable*** INTEGER.

***rotate_angle*** INTEGER.

***rtmp265_enable*** TEXT.

***srt_pass*** TEXT.

***srt_encrypt*** INTEGER.

***srt_mode*** INTEGER.

***srt_destAddr*** TEXT.

***mmulticastdest*** TEXT.

***mport*** TEXT.

***frame_loss_mode*** TEXT.

***frame_loss_gap*** INTEGER.

***frame_loss_overshot*** INTEGER.

***mounts*** TEXT.

***source_status_str*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Encoder response:
{"abitrate":"128000","acodec":"fdk_aaclc","aenable":"on","amulticastdest":"2
25.1.2.3","apair":0,"apid":120,"aport":"8700","aptspcr":250,"asamplerate":"1
","asource":"MIC","auth_passwd":"password","auth_user":"user","authonoff":"o
ff","auxonoff":"off","avmux_index":"streaming","channel":1,"enc_status":"RUN
NING","feccol":5,"feconoff":"off","fecrow":1,"filesize":"1024M","fprefix":"M
OV1_%C","frame_loss_gap":0,"frame_loss_mode":"none","frame_loss_overshot":10
0,"gps_overlay_char_size":"32","gps_overlay_device":"/dev/ttyS1","gps_overla
y_enable":"off","gps_overlay_location":"top_right","klvbrate":"1000","klvena
ble":"off","klvmode":"sdi","klvmuxmethod":"sync","klvpid":"35","klvserialbau
d":"115200","klvsrc":"/dev/ttyS1","lowdelay_opt":"off","mmulticastdest":"225
.1.2.3","mounts":"/dev/mmcblk0p1
/media/mmcblk0p1/,","mport":"8800","nfstrength":"0","pcrinterval":50,"pcrpid
":521,"pipenable":"off","piplocation":"top_right","pmtpid":31,"preset":"actv
_preset","ret":"0","rotate_angle":0,"rotate_enable":"off","rtmp265_enable":"
off","rtsp_auth_enable":"off","rtsp_auth_password":"admin","rtsp_auth_userna
me":"admin","rtsp_transport_mode":"all","source_status_str":" CAMERA
3840x2160p 29.97
fps\n","srt_destAddr":"192.168.0.6","srt_encrypt":0,"srt_mode":0,"srt_pass":
"password1234","storage":"/media/sda1","telopcharsize":"32","telopenable":"o
ff","teloplocation":"top_left","telopoutlinecolor":"0xFF000000","telopoutlin
eenable":"off","teloptext":"Z3Cam","teloptextcolor":"0xFFFFFFFF","tslowlat":
"on","tsrate":"3000K","vbitrate":"6M","vcodec":"h265","vcrop_enable":"off","
vcrop_height":1080,"vcrop_width":1920,"vcrop_x":0,"vcrop_y":0,"vcropaspect":
"off","vdelay":1000,"vdest":"192.168.0.6:8600","vframeratediv":1,"vgdr":"off
","vgopsize":60,"vinterlacemode":"combine","vmulticastdest":"225.1.2.3","vpi
d":221,"vprofile":"high","vprotocol":"rtsp","vquality":"balanced","vratectrl
":"vbr","vres":"follow_input","vsource":"CAMERA","zixiauthen":"off","zixifec
block":"50","zixilatency":"500","zixioverhead":"15","zixirateadjen":"on","zi
xisession":"test","zixiuser":"user"}
```


---

## filepicker

---

Get the file type and its information from the removable storage media.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***filepicker_filename*** TEXT,

***filepicker_size*** INTEGER.

***filepicker_lastmodified*** TEXT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Filepicket response:
{"data":[],"metadata":[{"datatype":"text","editable":false,"label":"File
Name","name":"filepicker_filename"},{"datatype":"integer","editable":false,"
label":"File
Size","name":"filepicker_size"},{"datatype":"text","editable":false,"label":
"Last Modified","name":"filepicker_lastmodified"}],"ret":"0"}
```

## fmt

Get the removable storage infromation from the device using fdisk command.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mounts*** TEXT,

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Format response: {"mounts":"mmcblk0p1\n
/dev/mmcblk0,","ret":"0","store_logs":"off"}
```

---

## focusStepSize

---

Get the current camera focus step values from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***focus_step_size*** INTEGER,

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Focus Step Size response: {"ret":"0","size":1}
```

## fpga

Get the current fpga from firmware filename.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***fpgafileglob*** TEXT,

***options*** TEXT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
FPGA response:
{"fpgafileglob":"fpga_none.bin|fpga2_0139_cvbs.bin","fpgaoptions":{"Boson
only        ":"fpga_unused.bin|fpga2_0139_boson.bin","Composite only
":"fpga_unused.bin|fpga2_0139_cvbs.bin","HDMI 4K| Boson
":"fpga_none.bin|fpga2_0139_boson.bin","HDMI 4K|
Composite":"fpga_none.bin|fpga2_0139_cvbs.bin","HDMI 4K| SonyLVDS
":"fpga_none.bin|fpga2_0139.bin","HDMI 4K|
```

```
Tamarisk":"fpga_none.bin|fpga2_0139_tamarisk.bin","HDMI 4K| Tau2
":"fpga_none.bin|fpga2_0139_tau2.bin","HDMI 4K| Tenum
":"fpga_none.bin|fpga2_0139_tenum.bin","HDMI 4K| Unused
":"fpga_none.bin|fpga2_none.bin","Micro-HDMI| Boson
":"fpga_none_microhdmi.bin|fpga2_0139_boson.bin","Micro-HDMI|
Composite":"fpga_none_microhdmi.bin|fpga2_0139_cvbs.bin","Micro-HDMI|
SonyLVDS ":"fpga_none_microhdmi.bin|fpga2_0139.bin","Micro-HDMI| Tamarisk
":"fpga_none_microhdmi.bin|fpga2_0139_tamarisk.bin","Micro-HDMI| Tau2
":"fpga_none_microhdmi.bin|fpga2_0139_tau2.bin","Micro-HDMI| Tenum
":"fpga_none_microhdmi.bin|fpga2_0139_tenum.bin","Micro-HDMI| Unused
":"fpga_none_microhdmi.bin|fpga2_none.bin","SonyLVDS only
":"fpga_unused.bin|fpga2_0139.bin","Tamarisk only
":"fpga_unused.bin|fpga2_0139_tamarisk.bin","Tau2  only
":"fpga_unused.bin|fpga2_0139_tau2.bin","Tenum only
":"fpga_unused.bin|fpga2_0139_tenum.bin"},"ret":"0"}
```


---

## history

---

Get the decoder history infromation from the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***channel*** TEXT,

***histidx*** TEXT.

***url*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
```

```
admin
```

---

**Example output:**

```
History response: {"ret":"0","status":"OK"}
```

---

## ipinfo

---

Get the internet protocol infromation from the board.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***local_ip*** TEXT,

***local_netmask*** TEXT.

***default_gw*** TEXT.

***ipmtu*** TEXT.

***eth_speed*** TEXT.

***eth_duplex*** TEXT.

***do_autostart*** BOOLEAN.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
IP Info response:
{"default_gw":"192.168.10.1","do_autostart":1,"eth_duplex":"AUTO","eth_speed
":"AUTO","ipmtu":1500,"local_dnsip":"8.8.8.8","local_dnsip2":"8.8.4.4","loca
l_hostname":"z3-
he4k","local_ip":"192.168.10.127","local_netmask":"255.255.255.0","ret":"0",
"use_dhcp":"1"}
```

## onvif

Get the ONVIF miscallenous settings from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***list_all_video_sources*** INTEGER,

***dns_from_dhcp*** INTEGER,

***ntp_from_dhcp*** INTEGER,

***ntp_enable*** INTEGER,

***daylightsaving*** INTEGER,

***timezone*** TEXT.

***http_enable*** INTEGER,

***http_port*** INTEGER,

***https_enable*** INTEGER,

***https_port*** INTEGER,

***rtsp_enable*** INTEGER,

***rtsp_port*** INTEGER,

***discoverable*** INTEGER,

***ntp*** TEXT.

***dns_search*** TEXT.

***ptz_timeout*** TEXT.

***primary fixed_profile_max*** INTEGER,

***secondary fixed_profile_max*** INTEGER,

***en_persistent_audio*** TEXT.

***onvif_enable*** BOOLEAN.

***vmd_enable*** BOOLEAN.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
ONVIF response:
{"daylightsaving":"0","discoverable":"1","dns_from_dhcp":"1","dns_search":"l
ocaldomain","en_persistent_audio":"1","fixed_profile_max":"1","http_enable":
"1","http_port":"80","https_enable":"0","https_port":"443","list_all_video_s
ources":"0","ntp":"pool.ntp.org","ntp_enable":"1","ntp_from_dhcp":"0","onvif
_enable":"on","ptz_timeout":10,"ret":"0","rtsp_enable":"1","rtsp_port":"554"
,"timezone":"TaipeiStandardTime-8","vmd_enable":"off"}
```

---

## onvif_vmd

---

Get the ONVIF video motion detection and too darkness settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***vmd_sens*** INTEGER,

***vmd_zone_modify*** TEXT.

***vmd_vcrop_width*** INTEGER.

***vmd_vcrop_height*** INTEGER.

***vmd_vcrop_x*** INTEGER.

***vmd_vcrop_y*** INTEGER.

***td_enable*** TEXT.

***td_thresh*** INTEGER.

***vmd_td_channel*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
ONVIF VMD response:
{"ret":"0","td_enable":"off","td_thresh":85,"vmd_enable":"off","vmd_sens":1,
"vmd_td_channel":"CH1","vmd_vcrop_height":0,"vmd_vcrop_width":0,"vmd_vcrop_x
":0,"vmd_vcrop_y":0,"vmd_zone_modify":"off"}
```

---

## permission

Get the current permission settings for admin, operators, and users from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***username*** TEXT,

***userlevel*** TEXT.

***api_command*** TEXT.

***superadmin*** TEXT.

**admin** TEXT.

***operator1*** TEXT.

***operator2*** TEXT.

***user1*** TEXT.

***user2*** TEXT.

***user3*** TEXT.

***anonymous*** TEXT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
PERMISSIONS response:
{"permissions":{"Z38021xImportHandler:post":"W","Z3DebugHandler:get":"R","Z3
```

DownloadMediaHandler:get":"R","Z3ExportHandler:get":"R","Z3FWUploadHandler:post":"W","Z3ImportHandler:post":"W","Z3LogoHandler:get":"?","Z3LogsHandler:get":"R","Z3RequestHandler:get:8021x":"?","Z3RequestHandler:get:GetCronJobs":"R","Z3RequestHandler:get:GetNFSMount":"?","Z3RequestHandler:get:aspect_info":"?","Z3RequestHandler:get:boardinfo":"R","Z3RequestHandler:get:cam_state":"R","Z3RequestHandler:get:camera":"?","Z3RequestHandler:get:camera_exposure":"?","Z3RequestHandler:get:camera_monitor_mode":"?","Z3RequestHandler:get:dec":"R","Z3RequestHandler:get:download":"?","Z3RequestHandler:get:drs_tamarisk_settings":"?","Z3RequestHandler:get:enc":"R","Z3RequestHandler:get:filepicker":"?","Z3RequestHandler:get:fmt":"W","Z3RequestHandler:get:focusStepSize":"?","Z3RequestHandler:get:fpga":"?","Z3RequestHandler:get:history":"R","Z3RequestHandler:get:ipinfo":"R","Z3RequestHandler:get:onvif":"?","Z3RequestHandler:get:onvif_vmd":"?","Z3RequestHandler:get:permission":"R","Z3RequestHandler:get:ptz":"?","Z3RequestHandler:get:ptz_preset":"?","Z3RequestHandler:get:ptz_tour":"?","Z3RequestHandler:get:snmp":"?","Z3RequestHandler:get:ssl":"?","Z3RequestHandler:get:stats":"?","Z3RequestHandler:get:sys":"R","Z3RequestHandler:get:term":"?","Z3RequestHandler:get:users":"R","Z3RequestHandler:get:video_group":"?","Z3RequestHandler:get:wifi":"?","Z3RequestHandler:get:zoomStepSize":"?","Z3RequestHandler:post:AStreamStatus":"?","Z3RequestHandler:post:AddChannel":"W","Z3RequestHandler:post:CameraControl:SetColorPallette":"W","Z3RequestHandler:post:CameraControl:ae_mode":"W","Z3RequestHandler:post:CameraControl:ae_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:agc_filter":"W","Z3RequestHandler:post:CameraControl:agc_midpoint":"W","Z3RequestHandler:post:CameraControl:agc_type":"W","Z3RequestHandler:post:CameraControl:auto_icr_disable":"W","Z3RequestHandler:post:CameraControl:auto_icr_enable":"W","Z3RequestHandler:post:CameraControl:autocal_disable":"W","Z3RequestHandler:post:CameraControl:autocal_enable":"W","Z3RequestHandler:post:CameraControl:backlight_inquiry":"R","Z3RequestHandler:post:CameraControl:backlight_off":"W","Z3RequestHandler:post:CameraControl:backlight_on":"W","Z3RequestHandler:post:CameraControl:black_hot":"W","Z3RequestHandler:post:CameraControl:brightness":"W","Z3RequestHandler:post:CameraControl:brightness_bias":"W","Z3RequestHandler:post:CameraControl:brightness_inquiry":"R","Z3RequestHandler:post:CameraControl:cam_control_inquiry":"R","Z3RequestHandler:post:CameraControl:cam_custom_recall":"W","Z3RequestHandler:post:CameraControl:cam_custom_reset":"W","Z3RequestHandler:post:CameraControl:cam_custom_set":"W","Z3RequestHandler:post:CameraControl:chroma_get_suppress":"R","Z3RequestHandler:post:CameraControl:chroma_suppress":"W","Z3RequestHandler:post:CameraControl:color_disable":"W","Z3RequestHandler:post:CameraControl:color_enable":"W","Z3RequestHandler:post:CameraControl:color_gain":"W","Z3RequestHandler:post:CameraControl:color_get_gain":"R","Z3RequestHandler:post:CameraControl:color_get_hue":"R","Z3RequestHandler:post:CameraControl:color_hue":"W","Z3RequestHandler:post:CameraControl:color_select":"W","Z3RequestHandler:post:CameraControl:colorlut_arctic":"W","Z3RequestHandler:post:CameraControl:colorlut_blackhot":"W","Z3RequestHandler:post:CameraControl:colorlut_disable":"W","Z3RequestHandler:post:CameraControl:colorlut_enable":"W","Z3RequestHandler:post:CameraControl:colorlut_globow":"W","Z3RequestHandler:post:CameraControl:colorlut_gradedfire":"W","Z3RequestHandler:post:CameraControl:colorlut_hottest":"W","Z3RequestHandler:post:CameraControl:colorlut_ironbow":"W","Z3RequestHandler:post:CameraControl:colorlut_lava":"W","Z3RequestHandler:post:CameraControl:colorlut_rainbow":"W","Z3RequestHandler:post:CameraControl:colorlut_rainbow_hc":"W","Z3RequestHandler:post:CameraControl:colorlut_whitehot":"W","Z3Reque

stHandler:post:CameraControl:contrast_adjust_level":"W","Z3RequestHandler:post:CameraControl:contrast_adjust_level_inquiry":"R","Z3RequestHandler:post:CameraControl:contrast_inquiry":"R","Z3RequestHandler:post:CameraControl:defog_inquiry":"R","Z3RequestHandler:post:CameraControl:defog_on_direct":"W","Z3RequestHandler:post:CameraControl:display_off":"W","Z3RequestHandler:post:CameraControl:display_on":"W","Z3RequestHandler:post:CameraControl:display_on_off":"W","Z3RequestHandler:post:CameraControl:do_ffc":"W","Z3RequestHandler:post:CameraControl:dvo_set_output_format_ycbcr":"W","Z3RequestHandler:post:CameraControl:dzoom_combine_mode":"W","Z3RequestHandler:post:CameraControl:dzoom_direct":"W","Z3RequestHandler:post:CameraControl:dzoom_inquiry":"R","Z3RequestHandler:post:CameraControl:dzoom_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:dzoom_off":"W","Z3RequestHandler:post:CameraControl:dzoom_on":"W","Z3RequestHandler:post:CameraControl:dzoom_separage_mode":"W","Z3RequestHandler:post:CameraControl:dzoom_separate_mode":"W","Z3RequestHandler:post:CameraControl:dzoom_stop":"W","Z3RequestHandler:post:CameraControl:dzoom_super_res":"W","Z3RequestHandler:post:CameraControl:dzoom_tele_var":"W","Z3RequestHandler:post:CameraControl:dzoom_wide_var":"W","Z3RequestHandler:post:CameraControl:dzoom_x1_max":"W","Z3RequestHandler:post:CameraControl:e_pan_tilt_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:e_pan_tilt_off":"W","Z3RequestHandler:post:CameraControl:e_pan_tilt_on":"W","Z3RequestHandler:post:CameraControl:eflip_inquiry":"R","Z3RequestHandler:post:CameraControl:eflip_off":"W","Z3RequestHandler:post:CameraControl:eflip_on":"W","Z3RequestHandler:post:CameraControl:expcomp_direct":"W","Z3RequestHandler:post:CameraControl:expcomp_down":"W","Z3RequestHandler:post:CameraControl:expcomp_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:expcomp_off":"W","Z3RequestHandler:post:CameraControl:expcomp_on":"W","Z3RequestHandler:post:CameraControl:expcomp_pos_inquiry":"R","Z3RequestHandler:post:CameraControl:expcomp_reset":"W","Z3RequestHandler:post:CameraControl:expcomp_up":"W","Z3RequestHandler:post:CameraControl:ffc_period":"W","Z3RequestHandler:post:CameraControl:ffc_temp_delta":"W","Z3RequestHandler:post:CameraControl:ffc_warn_time":"W","Z3RequestHandler:post:CameraControl:flicker_detection_inquiry":"R","Z3RequestHandler:post:CameraControl:flicker_reduction_inquiry":"R","Z3RequestHandler:post:CameraControl:flicker_reduction_off":"W","Z3RequestHandler:post:CameraControl:flicker_reduction_on":"W","Z3RequestHandler:post:CameraControl:focus_auto":"W","Z3RequestHandler:post:CameraControl:focus_direct":"W","Z3RequestHandler:post:CameraControl:focus_get_mode":"R","Z3RequestHandler:post:CameraControl:focus_get_pos":"R","Z3RequestHandler:post:CameraControl:focus_get_pos_meter":"R","Z3RequestHandler:post:CameraControl:focus_manual":"W","Z3RequestHandler:post:CameraControl:focus_near_limit":"W","Z3RequestHandler:post:CameraControl:focus_near_limit_inquiry":"R","Z3RequestHandler:post:CameraControl:focus_set_pos_meter":"W","Z3RequestHandler:post:CameraControl:focus_stop":"W","Z3RequestHandler:post:CameraControl:focus_tele_std":"W","Z3RequestHandler:post:CameraControl:focus_tele_var":"W","Z3RequestHandler:post:CameraControl:focus_toggle":"W","Z3RequestHandler:post:CameraControl:focus_wide_std":"W","Z3RequestHandler:post:CameraControl:focus_wide_var":"W","Z3RequestHandler:post:CameraControl:fpa_temp_inquiry":"R","Z3RequestHandler:post:CameraControl:gain":"W","Z3RequestHandler:post:CameraControl:gain_inquiry":"R","Z3RequestHandler:post:CameraControl:gain_mode":"W","Z3RequestHandler:post:CameraControl:get_camera_framerate":"R","Z3RequestHandler:post:CameraControl:get_camera_part_number":"R","Z3RequestHandler:post:CameraControl:get_camera_serial_number":"R","Z3RequestHandler:post:CameraControl:get_color_tables":"R","Z3

RequestHandler:post:CameraControl:get_colorlut_current":"R","Z3RequestHandler:post:CameraControl:get_colorlut_state":"R","Z3RequestHandler:post:CameraControl:get_fpa_temp_celsius":"R","Z3RequestHandler:post:CameraControl:get_low_delay_mode":"R","Z3RequestHandler:post:CameraControl:get_revision":"R","Z3RequestHandler:post:CameraControl:get_sensor_part_number":"R","Z3RequestHandler:post:CameraControl:get_sensor_serial_number":"R","Z3RequestHandler:post:CameraControl:get_software_rev":"R","Z3RequestHandler:post:CameraControl:get_tnr_state":"R","Z3RequestHandler:post:CameraControl:get_zoom_limits_mm":"R","Z3RequestHandler:post:CameraControl:high_sensitivity_inquiry":"R","Z3RequestHandler:post:CameraControl:high_sensitivity_off":"W","Z3RequestHandler:post:CameraControl:high_sensitivity_on":"W","Z3RequestHandler:post:CameraControl:hlc":"W","Z3RequestHandler:post:CameraControl:hlc_inquiry":"R","Z3RequestHandler:post:CameraControl:hr_inquiry":"R","Z3RequestHandler:post:CameraControl:hr_off":"W","Z3RequestHandler:post:CameraControl:hr_on":"W","Z3RequestHandler:post:CameraControl:ice_disable":"W","Z3RequestHandler:post:CameraControl:ice_enable":"W","Z3RequestHandler:post:CameraControl:icr_auto_inquiry":"R","Z3RequestHandler:post:CameraControl:icr_mode":"W","Z3RequestHandler:post:CameraControl:icr_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:icr_threshold_inquiry":"R","Z3RequestHandler:post:CameraControl:image_freeze_inquiry":"R","Z3RequestHandler:post:CameraControl:image_freeze_off":"W","Z3RequestHandler:post:CameraControl:image_freeze_on":"W","Z3RequestHandler:post:CameraControl:img_blackwhite_inquiry":"R","Z3RequestHandler:post:CameraControl:img_blackwhite_off":"W","Z3RequestHandler:post:CameraControl:img_blackwhite_on":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_hold":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_inquiry":"R","Z3RequestHandler:post:CameraControl:img_stabilizer_level":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_level_inquiry":"R","Z3RequestHandler:post:CameraControl:img_stabilizer_off":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_on":"W","Z3RequestHandler:post:CameraControl:iris":"W","Z3RequestHandler:post:CameraControl:iris_inquiry":"R","Z3RequestHandler:post:CameraControl:lens_control_inquiry":"R","Z3RequestHandler:post:CameraControl:lens_get_temp":"R","Z3RequestHandler:post:CameraControl:lr_reverse_inquiry":"R","Z3RequestHandler:post:CameraControl:lr_reverse_off":"W","Z3RequestHandler:post:CameraControl:lr_reverse_on":"W","Z3RequestHandler:post:CameraControl:min_shutter_inquiry":"R","Z3RequestHandler:post:CameraControl:min_shutter_limit":"W","Z3RequestHandler:post:CameraControl:min_shutter_limit_inquiry":"R","Z3RequestHandler:post:CameraControl:min_shutter_off":"W","Z3RequestHandler:post:CameraControl:min_shutter_on":"W","Z3RequestHandler:post:CameraControl:mute_on_off":"W","Z3RequestHandler:post:CameraControl:noise_2d3d":"W","Z3RequestHandler:post:CameraControl:noise_2d3d_inquiry":"R","Z3RequestHandler:post:CameraControl:noise_reduction":"W","Z3RequestHandler:post:CameraControl:noise_reduction_inquiry":"R","Z3RequestHandler:post:CameraControl:part_number_inquiry":"R","Z3RequestHandler:post:CameraControl:power_off":"W","Z3RequestHandler:post:CameraControl:power_on":"W","Z3RequestHandler:post:CameraControl:raw":"W","Z3RequestHandler:post:CameraControl:reboot_camera":"W","Z3RequestHandler:post:CameraControl:register_read":"R","Z3RequestHandler:post:CameraControl:register_write":"W","Z3RequestHandler:post:CameraControl:run_ffc":"W","Z3RequestHandler:post:CameraControl:serial_number_inquiry":"R","Z3RequestHandler:post:CameraControl:set_low_delay_mode":"W","Z3RequestHandler:post:CameraControl:set_monitor_mode":"W","Z3RequestHandler:post:CameraControl:shutter":"W","Z3RequestHandler:post:CameraControl:shutter_inquiry":"R","Z3RequestHandler:post:C

ameraControl:shutter_position":"R","Z3RequestHandler:post:CameraControl:shutter_temp":"R","Z3RequestHandler:post:CameraControl:shutter_temp_inquiry":"R","Z3RequestHandler:post:CameraControl:slow_shutter":"W","Z3RequestHandler:post:CameraControl:slow_shutter_inquiry":"R","Z3RequestHandler:post:CameraControl:slow_shutter_limit_inquiry":"R","Z3RequestHandler:post:CameraControl:spot_display":"W","Z3RequestHandler:post:CameraControl:spot_display_inquiry":"W","Z3RequestHandler:post:CameraControl:spot_meter_data_advanced":"W","Z3RequestHandler:post:CameraControl:spot_meter_data_inquiry":"R","Z3RequestHandler:post:CameraControl:spot_meter_inquiry":"R","Z3RequestHandler:post:CameraControl:spot_meter_mode":"W","Z3RequestHandler:post:CameraControl:spot_meter_pos_inquiry":"R","Z3RequestHandler:post:CameraControl:stable_zoom_inquiry":"R","Z3RequestHandler:post:CameraControl:stable_zoom_off":"W","Z3RequestHandler:post:CameraControl:stable_zoom_on":"W","Z3RequestHandler:post:CameraControl:tnr_disable":"W","Z3RequestHandler:post:CameraControl:tnr_enable":"W","Z3RequestHandler:post:CameraControl:version":"R","Z3RequestHandler:post:CameraControl:version_inquiry":"R","Z3RequestHandler:post:CameraControl:visibility_enhancer_inquiry":"R","Z3RequestHandler:post:CameraControl:visibility_enhancer_off":"W","Z3RequestHandler:post:CameraControl:visibility_enhancer_on":"W","Z3RequestHandler:post:CameraControl:visibility_enhancer_params":"W","Z3RequestHandler:post:CameraControl:visibility_enhancer_params_inquiry":"W","Z3RequestHandler:post:CameraControl:wb_auto_mode":"W","Z3RequestHandler:post:CameraControl:wb_auto_trace_mode":"W","Z3RequestHandler:post:CameraControl:wb_autotrace_mode":"W","Z3RequestHandler:post:CameraControl:wb_get_bgain":"R","Z3RequestHandler:post:CameraControl:wb_get_mode":"R","Z3RequestHandler:post:CameraControl:wb_get_mode_name":"R","Z3RequestHandler:post:CameraControl:wb_get_rgain":"R","Z3RequestHandler:post:CameraControl:wb_indoor_mode":"W","Z3RequestHandler:post:CameraControl:wb_manual_bgain_direct":"W","Z3RequestHandler:post:CameraControl:wb_manual_bgain_reset":"W","Z3RequestHandler:post:CameraControl:wb_manual_mode":"W","Z3RequestHandler:post:CameraControl:wb_manual_rgain_direct":"W","Z3RequestHandler:post:CameraControl:wb_manual_rgain_reset":"W","Z3RequestHandler:post:CameraControl:wb_onepush_mode":"W","Z3RequestHandler:post:CameraControl:wb_onepush_trigger":"W","Z3RequestHandler:post:CameraControl:wb_outdoor_auto_mode":"W","Z3RequestHandler:post:CameraControl:wb_outdoor_mode":"W","Z3RequestHandler:post:CameraControl:wb_sodium_lamp_auto_mode":"W","Z3RequestHandler:post:CameraControl:wb_sodium_lamp_fixed_mode":"W","Z3RequestHandler:post:CameraControl:wb_sodium_lamp_outdoor_mode":"W","Z3RequestHandler:post:CameraControl:white_hot":"W","Z3RequestHandler:post:CameraControl:wide_dynamic_range_inquiry":"R","Z3RequestHandler:post:CameraControl:wide_dynamic_range_off":"W","Z3RequestHandler:post:CameraControl:wide_dynamic_range_on":"W","Z3RequestHandler:post:CameraControl:zoom_direct":"W","Z3RequestHandler:post:CameraControl:zoom_get_focal_length_mm":"R","Z3RequestHandler:post:CameraControl:zoom_get_pos":"R","Z3RequestHandler:post:CameraControl:zoom_stop":"W","Z3RequestHandler:post:CameraControl:zoom_tele_std":"W","Z3RequestHandler:post:CameraControl:zoom_tele_var":"W","Z3RequestHandler:post:CameraControl:zoom_wide_std":"W","Z3RequestHandler:post:CameraControl:zoom_wide_var":"W","Z3RequestHandler:post:DecoderStatus":"?","Z3RequestHandler:post:DeleteChannel":"W","Z3RequestHandler:post:DeletePreset":"W","Z3RequestHandler:post:DisplayStatus":"?","Z3RequestHandler:post:Dynamic:analog_gain_db":"?","Z3RequestHandler:post:Dynamic:crop":"?","Z3RequestHandler:post:Dynamic:gop":"?","Z3RequestHandler:post:Dynamic:nfstrength":"?","Z3RequestHandler:post:Dynamic:pip_enable":"?","Z3RequestHandler:post:Dynamic:pip_location":"?","Z

3RequestHandler:post:Dynamic:rotate_angle":"?","Z3RequestHandler:post:Dynamic:rotate_center_offset":"?","Z3RequestHandler:post:Dynamic:rotate_enable":"?","Z3RequestHandler:post:Dynamic:rotate_mode":"?","Z3RequestHandler:post:Dynamic:startmulticast":"?","Z3RequestHandler:post:Dynamic:stopmulticast":"?","Z3RequestHandler:post:Dynamic:telop_text":"?","Z3RequestHandler:post:Dynamic:translate_enable":"?","Z3RequestHandler:post:Dynamic:translate_offset":"?","Z3RequestHandler:post:Dynamic:vconflict":"?","Z3RequestHandler:post:Dynamic:vrate":"?","Z3RequestHandler:post:Dynamic:vratediv":"?","Z3RequestHandler:post:EjectStorage":"W","Z3RequestHandler:post:EncoderStatus":"?","Z3RequestHandler:post:ErasePresets":"W","Z3RequestHandler:post:ExportPresets":"W","Z3RequestHandler:post:FactoryReset":"W","Z3RequestHandler:post:GetCameraLink":"?","Z3RequestHandler:post:GetCameraTab":"?","Z3RequestHandler:post:GetStatus":"?","Z3RequestHandler:post:GetVideoGroup":"?","Z3RequestHandler:post:GetWifiPass":"?","Z3RequestHandler:post:InsQuery":"W","Z3RequestHandler:post:LoadUser":"W","Z3RequestHandler:post:Login":"W","Z3RequestHandler:post:Logout":"W","Z3RequestHandler:post:Overlay":"?","Z3RequestHandler:post:OverlayStop":"W","Z3RequestHandler:post:PtzAbsoluteMove":"W","Z3RequestHandler:post:PtzAuxiliary":"W","Z3RequestHandler:post:PtzClearPreset":"W","Z3RequestHandler:post:PtzContinuousMove":"W","Z3RequestHandler:post:PtzGotoPreset":"W","Z3RequestHandler:post:PtzNewTourSpot":"W","Z3RequestHandler:post:PtzPosition":"W","Z3RequestHandler:post:PtzPreset":"W","Z3RequestHandler:post:PtzQuery":"W","Z3RequestHandler:post:PtzRemoveTourSpot":"W","Z3RequestHandler:post:PtzSetPreset":"W","Z3RequestHandler:post:PtzStartTour":"W","Z3RequestHandler:post:PtzStop":"W","Z3RequestHandler:post:RemoveJobs":"W","Z3RequestHandler:post:RestartBoard":"W","Z3RequestHandler:post:SaveCamera":"?","Z3RequestHandler:post:SaveCameraBoson":"?","Z3RequestHandler:post:SaveCameraDRSTamarisk":"?","Z3RequestHandler:post:SaveCameraLink":"?","Z3RequestHandler:post:SaveCameraLx":"?","Z3RequestHandler:post:SaveCameraTau":"?","Z3RequestHandler:post:SaveJobs":"W","Z3RequestHandler:post:SavePresets":"W","Z3RequestHandler:post:SaveUser":"W","Z3RequestHandler:post:SetAPConfig":"?","Z3RequestHandler:post:SetAdv":"?","Z3RequestHandler:post:SetAudio":"W","Z3RequestHandler:post:SetCameraLink":"W","Z3RequestHandler:post:SetConsole":"?","Z3RequestHandler:post:SetDDNSEnable":"W","Z3RequestHandler:post:SetDSCP":"?","Z3RequestHandler:post:SetDebugLevel":"W","Z3RequestHandler:post:SetDevName":"W","Z3RequestHandler:post:SetDisplay":"?","Z3RequestHandler:post:SetEncoder":"W","Z3RequestHandler:post:SetEncoderCameraControl":"W","Z3RequestHandler:post:SetFpga":"W","Z3RequestHandler:post:SetIp":"W","Z3RequestHandler:post:SetLoginLimit":"W","Z3RequestHandler:post:SetNFS":"W","Z3RequestHandler:post:SetNMEAEnable":"?","Z3RequestHandler:post:SetOnvif":"W","Z3RequestHandler:post:SetOnvifVMD":"W","Z3RequestHandler:post:SetRMF":"W","Z3RequestHandler:post:SetRTSP":"W","Z3RequestHandler:post:SetSNTP":"W","Z3RequestHandler:post:SetStartLogs":"W","Z3RequestHandler:post:SetStopLogs":"W","Z3RequestHandler:post:SetTermSrvEnable":"?","Z3RequestHandler:post:SetViVpssMode":"?","Z3RequestHandler:post:SetVideoGroup":"W","Z3RequestHandler:post:SetViewport":"?","Z3RequestHandler:post:SetWifiAP":"?","Z3RequestHandler:post:SetZFinderEnable":"W","Z3RequestHandler:post:SourceStatus":"?","Z3RequestHandler:post:StartChannel":"W","Z3RequestHandler:post:StartMTS":"W","Z3RequestHandler:post:StopChannel":"W","Z3RequestHandler:post:StopMTS":"W","Z3RequestHandler:post:StreamStatus":"?","Z3RequestHandler:post:TempStatus":"?","Z3RequestHandler:post:UpdatePtz":"?","Z3RequestHandler:post:UpdatePtzPreset":"W","Z3RequestHandler:post:UpdatePtzTourSpots":"W","Z3RequestHandler:post:UpdateTerm":"W","Z3RequestHandler:post:UpdateTermFor

```
Fpga":"W","Z3RequestHandler:post:ViscaPort":"x","Z3RequestHandler:post:camera_inquiry":"?","Z3RequestHandler:post:fmt_media":"W","Z3RequestHandler:post:update_require_web_login":"W","Z3RequestHandler:post:user_add":"W","Z3RequestHandler:post:user_remove":"W","Z3RequestHandler:post:user_update_level":"W","Z3RequestHandler:post:user_update_password":"W","Z3SSLImportHandler:post":"W","Z3SnapshotHandler:get":"R","Z3TabHandler:get":"-","Z3UploadHandler:post":"W","Z3WebSocketHandler:get":"R"},"ret":"0","status":"OK","userlevel":0,"username":"admin"}
```


---

## ptz

---

Get the current PTZ settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***token*** TEXT,

***node*** TEXT.

***name*** TEXT.

***enabled*** BOOLEAN.

***pelcoaddr*** INTEGER.

***tcpport*** INTEGER.

***ptztype*** TEXT.

***flip*** TEXT.

***pos*** TEXT.

***maxtilt*** INTEGER.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
PTZ response:
{"data":[{"id":1,"values":{"db_videosource":"CAMERA","enabled":1,"flip":"non
e","maxtilt":90,"name":"videoin1","pelcoaddr":1,"pos":"simulated","ptztype":
"relative","rowid":1,"tcpport":2000,"token":"ptz"}},{"id":2,"values":{"db_vi
deosource":"CAMERA2","enabled":1,"flip":"none","maxtilt":90,"name":"videoin2
","pelcoaddr":2,"pos":"simulated","ptztype":"relative","rowid":2,"tcpport":2
000,"token":"ptz2"}}],"metadata":[{"datatype":"text","editable":false,"label
":"Video
Source","name":"db_videosource"},{"datatype":"text","editable":false,"label"
:"ONVIF
Token","name":"token"},{"datatype":"text","editable":true,"label":"ONVIF
Name","name":"name"},{"datatype":"bool","editable":true,"label":"Enabled","n
ame":"enabled"},{"datatype":"integer","editable":true,"label":"RS485
Address","name":"pelcoaddr"},{"datatype":"integer","editable":true,"label":"
TCP
Port","name":"tcpport"},{"datatype":"text","editable":true,"label":"Type","n
ame":"ptztype"},{"datatype":"text","editable":true,"label":"Flip","name":"fl
ip"},{"datatype":"text","editable":true,"label":"Position","name":"pos"},{"d
atatype":"integer","editable":true,"label":"Max
Tilt","name":"maxtilt"}],"ret":"0"}
```

---

## ptz_preset

---

Get the PTZ preseting values from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***profiletoken*** TEXT,

***token*** TEXT.

***name*** TEXT.

***pan*** FLOAT.

***tilt*** FLOAT.

***zoom*** FLOAT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
PTZ Preset response:
{"data":[{"id":1,"values":{"action":"blank","name":"home","profiletoken":"pr
ofile1","rowid":1,"token":"1"}}],"metadata":[{"datatype":"text","editable":f
alse,"label":"Profile","name":"profiletoken"},{"datatype":"text","editable":
false,"label":"PresetID","name":"token"},{"datatype":"text","editable":true,
"label":"Name","name":"name"},{"datatype":"html","editable":false,"label":"A
ction","name":"action"}],"ret":0}
```

---

## ptz_tour

---

Get the PTZ tour information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***ProfileToken*** TEXT,

***PresetTourToken*** TEXT.

***name*** TEXT.

***autostart*** BOOLEAN.

***RecurringTime*** INTEGER.

***RecurringDuration*** TEXT.

***PresetTourDirection*** INTEGER.

***RandomPresetOrder*** BOOLEAN.

***ProfileToken*** TEXT.

***PresetTourToken*** TEXT.

***RowIndex*** INTEGER.

***PresetDetailToken*** TEXT.

***PresetDetailHome*** BOOLEAN.

***StayTime*** TEXT.

***pan*** FLOAT.

***tilt*** FLOAT.

***zoom*** FLOAT.

***pspeed*** FLOAT.

***tspeed*** FLOAT.

***zspeed*** FLOAT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
PTZ tour response:
{"preset":{"data":[{"id":1,"values":{"name":"home","pan":0,"profiletoken":"p
```

rofile1","rowid":1,"tilt":0,"token":"1","zoom":0}}],"metadata":[{"datatype":"text","editable":true,"label":"Profiletoken","name":"profiletoken"},{"datatype":"text","editable":true,"label":"Token","name":"token"},{"datatype":"text","editable":true,"label":"Name","name":"name"},{"datatype":"real","editable":true,"label":"Pan","name":"pan"},{"datatype":"real","editable":true,"label":"Tilt","name":"tilt"},{"datatype":"real","editable":true,"label":"Zoom","name":"zoom"}]},"ret":0,"spot":{"data":[],"metadata":[{"datatype":"text","editable":false,"label":"Profile","name":"ProfileToken"},{"datatype":"text","editable":false,"label":"Tour","name":"PresetTourToken"},{"datatype":"text","editable":true,"label":"Preset","name":"PresetDetailToken"},{"datatype":"text","editable":true,"label":"StayTime","name":"StayTime"},{"datatype":"integer","editable":false,"label":"RowIndex","name":"RowIndex"},{"datatype":"html","editable":"false","label":"Action","name":"action"}]},"tour":{"data":[{"id":1,"values":{"PresetTourToken":"1","ProfileToken":"profile1","name":"tour1","rowid":1}},{"id":2,"values":{"PresetTourToken":"2","ProfileToken":"profile1","name":"tour2","rowid":2}},{"id":3,"values":{"PresetTourToken":"3","ProfileToken":"profile1","name":"tour3","rowid":3}},{"id":4,"values":{"PresetTourToken":"4","ProfileToken":"profile1","name":"tour4","rowid":4}},{"id":5,"values":{"PresetTourToken":"5","ProfileToken":"profile1","name":"tour5","rowid":5}},{"id":6,"values":{"PresetTourToken":"6","ProfileToken":"profile1","name":"tour6","rowid":6}},{"id":7,"values":{"PresetTourToken":"7","ProfileToken":"profile1","name":"tour7","rowid":7}},{"id":8,"values":{"PresetTourToken":"8","ProfileToken":"profile1","name":"tour8","rowid":8}},{"id":9,"values":{"PresetTourToken":"9","ProfileToken":"profile1","name":"tour9","rowid":9}},{"id":10,"values":{"PresetTourToken":"10","ProfileToken":"profile1","name":"tour10","rowid":10}},{"id":11,"values":{"PresetTourToken":"11","ProfileToken":"profile1","name":"tour11","rowid":11}},{"id":12,"values":{"PresetTourToken":"12","ProfileToken":"profile1","name":"tour12","rowid":12}},{"id":13,"values":{"PresetTourToken":"13","ProfileToken":"profile1","name":"tour13","rowid":13}},{"id":14,"values":{"PresetTourToken":"14","ProfileToken":"profile1","name":"tour14","rowid":14}},{"id":15,"values":{"PresetTourToken":"15","ProfileToken":"profile1","name":"tour15","rowid":15}},{"id":16,"values":{"PresetTourToken":"16","ProfileToken":"profile1","name":"tour16","rowid":16}}],"metadata":[{"datatype":"text","editable":false,"label":"profile1","name":"ProfileToken"},{"datatype":"text","editable":false,"label":"TourID","name":"PresetTourToken"},{"datatype":"text","editable":true,"label":"Name","name":"name"}]}}

---

## snmp

---

Get the Simple Network Management Protocol values from the Database

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***power_trap_en*** INTEGER,

***input_loss_trap_en*** INTEGER.

***input_recover_trap_en*** INTEGER.

***temp_high_trap_en*** INTEGER.

***temp_recover_trap_en*** INTEGER.

***high_temp*** INTEGER.

***nominal_temp*** INTEGER.

***high_temp_suppress*** INTEGER.

***nominal_temp_suppress*** INTEGER.

***trap_hosts*** INTEGER.

***snmp_v3_en*** INTEGER.

***snmp_v3_user*** TEXT.

***snmp_v3_passwd*** TEXT.

***snmp_v3_usrpro*** TEXT.

***snmp_v3_encrypt*** INTEGER.

***snmp_v3_encrypt_passwd*** TEXT.

***snmp_v3_encpro*** TEXT.

***snmp_v3_engineid*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
SNMP response:
{"high_temp":80,"high_temp_suppress":30,"input_loss_trap_en":0,"input_recove
r_trap_en":0,"nominal_temp":60,"nominal_temp_suppress":30,"power_trap_en":0,
"ret":"0","snmp_v3_en":0,"snmp_v3_encpro":"des","snmp_v3_encrypt":0,"snmp_v3
_encrypt_passwd":"z3password","snmp_v3_engineid":"","snmp_v3_passwd":"z3pass
word","snmp_v3_user":"z3user","snmp_v3_usrpro":"md5","temp_high_trap_en":0,"
temp_recover_trap_en":0,"trap_hosts":"192.168.0.6"}
```

---

## ssl

---

Get the Secure Socket Layer settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***https_enable*** TEXT,

***http_enable*** TEXT.

***https_port*** INTEGER.

***http_port*** INTEGER.

***cert_status*** TEXT.

***key_status*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
SSL response:
{"cert_status":"Unavailable","http_enable":"on","http_port":80,"https_enable
":"off","https_port":443,"key_status":"Unavailable","ret":"0"}
```

## stats

Get the available memory infromation from the device.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mem_free_kb*** TEXT,

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
STATUS response: {"astream_status_str":"\tChannel 1 Codec fdk_aaclc
Samplerate 48000 Input MIC Frames 546732 \tChannel 11 Codec fdk_aaclc
Samplerate 48000 Input MIC Frames 545860  OK","encoder_status_str":"   ***
Encode Bitstream Received Statistics ***    CH | Bitrate (Kbps) | Actual
Bitrate | FPS  | Actual FPS | Key-frame FPS | Width | Height   -------------
-------------------------------------------------------------------------
------------------   1 |        6000.00 |        6006.33 | 29.97 |    29.97
|           0.5 |  3840 |   2160  |
```

```
OK","mem_free_kb":670856,"ret":"0","source_status_str":" CAMERA 3840x2160p
29.97 fps\n","status":"OK","stream_status_str":"Channel 1 Codec H265-(HEVC)
URL rtsp Frames 348194  OK","temp_status_str":" LENS 29  CPU 38.0  OK"}
```


---

## sys

---

Get the current system settings from the Database and the ip information from the device.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***local_ip*** TEXT,

***local_netmask*** TEXT.

***preset*** TEXT.

***enc_channels*** TEXT.

***opmode*** TEXT.

***username*** TEXT.

***userlevel*** TEXT.

***do_autostart*** BOOLEAN.

***syspassword*** TEXT.

***session_id*** TEXT.

***disp_std*** TEXT.

***disp_std2*** TEXT.

***disp_input*** TEXT.

***disp_mode*** TEXT.

***disp_layout*** TEXT.

***enc_adv_setting*** TEXT.

***enc_vivpss_mode_enable*** TEXT.

***enable_sntp*** TEXT.

***sntp_servers*** TEXT.

***enable_snmp*** TEXT.

***sysdevicename*** TEXT.

***timezone*** TEXT.

***timezone_name*** TEXT.

***termserve_remote_enable*** TEXT.

***termserve_shared_enable*** TEXT.

***zfinder_enable*** TEXT.

***diff_serve*** INTEGER.

***enable_ptp*** TEXT.

***ptp_role*** TEXT.

***nmea_enable*** TEXT.

***nfs_enable*** TEXT.

***nfs_server*** TEXT.

***nfs_server_root*** TEXT.

***ddns_enable*** TEXT.

***ddns_provider*** TEXT.

***ddns_username*** TEXT.

***ddns_password*** TEXT.

***rtspd_port*** INTEGER.

***rtspd_timeout*** INTEGER.

***maxlogin*** INTEGER.

***login_window*** INTEGER.

***lockout_interval*** INTEGER.

***login_timeout*** INTEGER.

***require_web_login*** BOOLEAN.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
SYSTEM response:
{"ANALOG_analog_gain_db":7,"MICL_analog_gain_db":-12,"MIC_analog_gain_db":7,
"console_enable":"on","ddns_enable":"off","ddns_hostname":"","ddns_password"
:"","ddns_provider":"freedns","ddns_username":"","diff_serve":0,"disp_input"
:"primary","disp_layout":"1x1","disp_mode":"ultrahd","disp_std":"auto","disp
_std2":"auto","do_autostart":1,"enable_ptp":"false","enable_snmp":"false","e
nable_sntp":"true","enc_adv_setting":"off","enc_vivpss_mode_enable":"off","l
ocal_ip":"192.168.10.127","local_netmask":"255.255.255.0","lockout_interval"
:900,"login_timeout":900,"login_window":900,"maxlogin":3,"nfs_enable":"off",
"nfs_server":"192.168.1.6","nfs_server_root":"/c/media","nmea_enable":"off",
"pe0":"encoder","pe1":"","pe2":"","pe3":"","pe4":"","pe5":"","pe6":"","pe7":
"","pe8":"","ptp_role":"auto","require_web_login":0,"ret":"0","rtspd_port":5
54,"rtspd_timeout":60,"session_id":"---
","sntp_servers":"pool.ntp.org","sysdevicename":"Z3Cam","syspassword":"","te
rmserve_remote_enable":"on","termserve_shared_enable":"on","timezone":"CST6C
DT,M3.2.0,M11.1.0","timezone_name":"America/Chicago","userlevel":0,"username
":"admin","wifi_exists":false,"zfinder_enable":"on"}
```

---

## term

---

Get the remote terminal information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***term_mode*** TEXT,

***term_protocol*** TEXT.

***term_localport*** TEXT.

***term_servaddr*** TEXT.

***term_servport*** TEXT.

***term_data_bits*** TEXT.

***term_parity*** TEXT.

***term_stop_bits*** TEXT.

***term_baudrate*** TEXT.

***term_devicefile*** TEXT.

***term_function*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
TERM response:
{"data":[{"id":1,"values":{"rowid":1,"term_baudrate":"2400","term_data_bits"
:"8","term_devicefile":"/dev/ttyS4","term_function":"rs485","term_localport"
:"2000","term_parity":"None","term_stop_bits":"1"}},{"id":2,"values":{"rowid
":2,"term_baudrate":"9600","term_data_bits":"8","term_devicefile":"/dev/ttyS
2","term_function":"camera1_visca","term_localport":"1000","term_parity":"No
ne","term_stop_bits":"1"}},{"id":3,"values":{"rowid":3,"term_baudrate":"9600
","term_data_bits":"8","term_devicefile":"/dev/ttyS3","term_function":"off",
"term_localport":"1001","term_parity":"None","term_stop_bits":"1"}},{"id":4,
"values":{"rowid":4,"term_baudrate":"115200","term_data_bits":"8","term_devi
cefile":"/dev/ttyS1","term_function":"user","term_localport":"1500","term_pa
rity":"None","term_stop_bits":"1"}},{"id":5,"values":{"rowid":5,"term_baudra
te":"115200","term_data_bits":"8","term_devicefile":"/dev/ttyS0","term_funct
ion":"console","term_localport":"1800","term_parity":"None","term_stop_bits"
:"1"}}],"metadata":[{"datatype":"text","editable":true,"label":"Serial
Port","name":"term_devicefile"},{"datatype":"text","editable":true,"label":"
```

```
Baud
Rate","name":"term_baudrate"},{"datatype":"text","editable":true,"label":"Da
ta
Bits","name":"term_data_bits"},{"datatype":"text","editable":true,"label":"P
arity","name":"term_parity"},{"datatype":"text","editable":true,"label":"Sto
p
Bits","name":"term_stop_bits"},{"datatype":"text","editable":true,"label":"F
unction","name":"term_function"},{"datatype":"text","editable":true,"label":
"TCP Port","name":"term_localport"}],"ret":"0"}
```


---

## users

---

Get the ONVIF user information and the permission level from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***Name*** TEXT,

***Level*** INTEGER.

***Password*** TEXT.

***Password_MD5*** TEXT.

***Password_SHA256*** TEXT.

***Password_SHA512_256*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
USERS response:
{"data":[{"id":1,"values":{"Level":"0","Name":"admin","action":"blank","rowi
d":1}},{"id":2,"values":{"Level":"7","Name":"guest","action":"blank","rowid"
:2}}],"metadata":[{"datatype":"text","editable":false,"label":"Name","name":
"Name"},{"datatype":"text","editable":true,"label":"Level","name":"Level"},{
"datatype":"html","editable":false,"label":"Action","name":"action"}],"ret":
0}
```

---

## video_group

---

Get the current video group settings from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***vgroup*** INTEGER.

***crop_enable*** TEXT.

***crop_width*** INTEGER.

***crop_height*** INTEGER.

***crop_x*** INTEGER.

***crop_y*** INTEGER.

***nr_enable*** TEXT.

***pip_small_dim*** TEXT. Possible Values: 720×576, 720×480, 640×360, 352×288 or 352×240 (Qualcomm v6.01 or higher)

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Video Group response:
{"crop_enable":"off","crop_height":0,"crop_width":0,"crop_x":0,"crop_y":0,"n
r_enable":"off","ret":"0","vgroup":1}
```

## wifi

Get the Wifi details from the device and wifi client information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***networks*** TEXT,

***cur_network*** TEXT,

***cur_pass*** TEXT,

***ap_pass*** TEXT,

***ap_pass_en*** BOOLEAN.

***ap_ssid*** TEXT,

***wifi_client_local_ip*** TEXT,

***wifi_client_local_netmask*** TEXT.

***wifi_client_default_gw*** TEXT.

***wifi_client_use_dhcp*** BOOLEAN.

---

### zoomStepSize

---

Get the current camera zoom step values from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***zoom_step_size*** INTEGER,

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Zoom Step Size response: {"ret":"0","size":1}
```

---

# File Upload for MFR or UPD Image Control

## Upload Image (POST)

---

Upload the z3 image for e.g. (z3-30-0139-mfr.img or z3-30-0139-upd.img) in the z3 device.

Before uploading the new image, it checks, whether the device has enough free memory or not and removes the coredump files. If the image is *mfr.img then it removes the default database from the data partition. If its *upd.img then it will not remove the database values.

Note:
It takes approximately 5 minutes to update; DO NOT SEND ANY OTHER API Commands after uploading the image for at least 5 minutes. (time.sleep(300) or equivalent in scripts)

**Parameters:**

---

***img_file*** Get the file name which needs to be flashed in the z3 device.

**Returns:**

---

Flash the image and reboot the device.

Note:
It takes approximately 5 minutes to update; DO NOT SEND ANY OTHER API Commands after uploading the image for at least 5 minutes. (time.sleep(300) or equivalent in scripts)

---

# Download File and Remove File Control

See **download_remove** to remove files from the **Generic Actions (POST)** action

## Download Media (GET)

```
http://<ENCODER_IP>/download_media.cgi?mediafile=/location/filename
```

For e.g.

```
http://<ENCODER_IP>/download_media.cgi?mediafile=/media/sda1/MOV1_000001.mp4
```

Download the media file. If the output format is mp4 or TS file, then only z3 device allows downloading the files. Also provides the option for deleting the file.

**Parameters:**

***mediafile*** Get the media file name.

**Returns:**

JSON formatted table with values below.

ret Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***File Name*** Name of the download file name .

***File Size*** Size of the download file name.

***Last Modified*** Last modified the file date and time.

# download.html to get list of downloadable files

```
http://<ENCODER_IP>/download.html?chn=1
```

chn **=[1,2,3,4]**

List the download content. The content will be either .mp4 file format.

# Encoder Actions (POST)

## SetEncoder

Write encoder settings to active preset. See Python example code for Seting Configuration below

## EncoderStatus

Requests an update to encoder_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

### StreamStatus

Requests an update to stream_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

### AStreamStatus

Requests an update to astream_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

### SourceStatus

Requests an update to source_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

---

## Setting Encoder Configuration

To set the encode configuration per channel the **SetEncoder** action is used with a post. All variables should be present or they will be replaced with a defualt that may not be valid for your application. More than one channel can be set per command. It is recommended you first read the current settings from the encoder then modify them and send the new settings back as in the Python example below:

```
import requests, json, sys

server_url='http://192.168.0.120/cgi-bin/control.cgi'
channel = 1

enc_cfg = requests.get(server_url, params='ctrl=enc&chn={}'.format(channel))
print enc_cfg.json()

enc_cfg_json=enc_cfg.json()
enc_cfg_json['vframeratediv'] = 2 #set frame rate divider to 2
new_cfg = {}
new_cfg['action'] = 'SetEncoder'
for key, value in enc_cfg_json.iteritems():
print "key = {} value = {}".format(key,value)
new_idx = "enc_{}_{}".format(channel,key)
print "setting {} to {}".format(new_idx,value)
```

```
new_cfg[new_idx] = value

print new_cfg
requests.post(server_url, data=new_cfg)
```

# snapshot.cgi Control

```
http://<ENCODER_IP>/snapshot.cgi?size=3840x2160&quality=100

http://<ENCODER_IP>/snapshot.cgi?chn=1&size=1280x720&quality=80
```

valid options are:

quality **=[20-100]**

* *size **=[any supported valid resolution]**

chn **=[1,2,3,4]**

```
quality** =[0-100] supported resolutions are:
follow_input

**4K Models Only**
3840×2160
2560×1440
**4K and HD Models **
1920×1080
1440×1080
1280×1024
1280×720
1024×768
960×720
1024×576
800×600
720×576
704×576
720×480
640×512
640×480
640×360
352×576
420×380
352×288
352×240
336×256
320×240
320×180
```

# Reading Statistics

**Python example of updating and reading encoder statistics:**

```
import requests, json, sys
server_url='http://192.168.0.120/cgi-bin/control.cgi'

requests.post(server_url, {'action': 'EncoderStatus'} )
requests.post(server_url, {'action': 'StreamStatus'}  )
requests.post(server_url, {'action': 'AStreamStatus'} )
requests.post(server_url, {'action': 'SourceStatus'}  )
requests.post(server_url, {'action': 'TempStatus'}  )

print requests.get(server_url, params='ctrl=stats&chn=null').json()
```

**Example output:**

```
{u'status': u'OK',
u'astream_status_str': u'\tChannel 6 Codec fdk_aaclc Samplerate 48000 Input
MICL Frames 32287820  OK',
u'temp_status_str': u' LENS 29  CPU 78.700  FPGA 89.8  OK',
u'ret': u'0',
u'encoder_status_str': u'   *** Encode Bitstream Received Statistics ***
CH | Bitrate (Kbps) | Actual Bitrate | FPS  | Actual FPS | Key-frame FPS |
Width | Height   ------------------------------------------------------------
-----------------------------------------------
1 |        4000.00 |        4092.18 | 60.0 |     59.8 |           1.0 |
1920 |   1080  |  OK',
u'source_status_str': u' CAMERA 1920x1080p 60.00 fps\n',
u'stream_status_str': u'Channel 1 URL rtsp Frames 12883581  OK'}
```

# Authentication

By default, no authentication is required to access the HTTP API.

To enable authentication, go to the “**Users**” tab. Click on the “**Require WebUI User Password**” button.

The superadmin username will be “**admin**” with “**admin**” as the default password.

To change the default password, under **Action** click on the **Reset Password** icon.

The authentication method is HTTP Digest authentication. HTTP Basic authentication is not supported.

**Python example code of HTTP Digest authentication**

```
import requests,json, sys
```

```
from requests.auth import HTTPDigestAuth

auth=HTTPDigestAuth('admin','mypassword')

print requests.get(server_url, params='ctrl=stats&chn=null',
auth=auth).json()
```

# Appendix A: Sample Python Scripts

The sample python scripts below require Python3.x.

The requests package must be installed.

You can install using

1) Use the Ubuntu package manager

sudo apt install python3-requests

2) Use the Python3 native package installer

sudo pip3 install requests

## SaveUser (saveUser_DIGEST_ARGPARSE.py)

Save the state, preset, encoder/decoder settings for the user.

#### Python example code of HTTP Digest SaveUser

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
```

```
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
payload = "action=SaveUser&enc_current_preset=charles";
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   setip_respose response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -
passwd admin
```

**Example output:**

```
SaveUser response: {"ret":"0","status":"OK"}
```

# SetIp (set_ip_DIGEST_ARGPARSE.py)

Control networking settings of the Z3 device such as local IPv4 IP address, netmask, DNS Server primary, DNS Server secondary, etc. Additionally control Encoder Auto-Start after boot-up and enable/disable showing of advanced WebUI settings.

**Python example code of HTTP Digest SetIp**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-newip', help='Set this IP')
```

```
parser.add_argument('-mask', help='Subnet Mask',default='255.255.255.0')
parser.add_argument('-gateway', help='Network Gateway',default='172.30.1.1')
parser.add_argument('-dns1', help='DNS 1',default='8.8.8.8')
parser.add_argument('-dns2', help='DNS 2',default='8.8.4.4')
parser.add_argument('-ipmtu', help='MTU Speed',default=1500)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
payload =
"action=SetIp&local_ip=172.30.1.112&local_netmask=255.255.255.0&default_gw=1
72.30.1.1&local_dnsip=8.8.8.8&local_dnsip2=8.8.4.4&use_dhcp=no&ipmtu=1500ð_s
peed=AUTOð_duplex=AUTO&do_autostart=1&enc_adv_setting=off";
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   setip_respose response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 set_ip_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -passwd
admin
```

**Example output:**

```
dns1='8.8.8.8', dns2='8.8.4.4', gateway='172.30.1.1', ip='192.168.1.105',
ipmtu=1500, mask='255.255.255.0', newip=None, passwd='admin',
username='admin'
setip_respose response: {"ret":"0","status":"OK"}
```

# cam_state (sys_DIGEST_ARGPARSE.py)

Get the current camera state information from the Database.

### Python example code of HTTP Digest camera state

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
```

```
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}

payload='ctrl=cam_state&chn=null';
rs = http_request_get(payload)
print_immediately(
  "{:>6}: {}   Sys response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 sys_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -passwd
admin
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin')
Sys response:
{"camera1_if_type":"Sony_LVDS","camera2_if_type":"CVBS","enc_channels":"1,2"
,"enc_current_preset":"charles","fpgafileglob":"fpga_none.bin|fpga2_0139_cvb
s.bin","preview_auto_start":"1","ret":"0","visca_present":"true"}
```

# CameraControl for Boson camera (Zoom_Boson.py)

Send control commands to boson camera. Only one command can be sent at a time;

### Python example code of HTTP Digest Camera control for Boson camera

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
import sys
from requests.auth import HTTPDigestAuth

if sys.version_info[0] <3:
  raise Exception("Python 3 or a more recent version is required.")

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    # print(msg, flush=True)
    print(msg)
    sys.stdout.flush()

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
```

```
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)
if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address
# Set Zoom STEP
zooms=args.zs;
zoomstep=args.zoom_step;

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)
control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
# Boson min Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d00020000000000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
```

```
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson mid Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d00020000015000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson mid Zoom xB0 yA0
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d00020000015000000B0000000A0';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
```

```
   rs.text
  )
)
# Boson max Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d00020000030000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson no Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d00020000000000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

### Command to execute

```
python3 Zoom_Boson.py -ip 172.28.30.103
```

**Example output:**

```
(ip='172.28.30.103', passwd='admin', username=None, zoom_step=False, zs='3')
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000000000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000015000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000015000000B0000000A0","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000030000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
```

# CameraControl for Sony Visca camera (Zoom_visca.py)

Send control commands to visca camera. Only one command can be sent at a time;

**Python example code of HTTP Digest Camera control for Visca camera**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
import sys
from requests.auth import HTTPDigestAuth

if sys.version_info[0] <3:
  raise Exception("Python 3 or a more recent version is required.")

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    # print(msg, flush=True)
    print(msg)
    sys.stdout.flush()

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
```

```
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
parser.add_argument('-z', help='Zoom STEP level 0 - 7; Default
3',default='3')
parser.add_argument('-zs', help='Do Zoom STEP defined using -z, else normal
zoom',action='store_true')
args = parser.parse_args()
print(args)
if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address
# Set Zoom STEP
zooms=args.z;
zoomstep=args.zs;

if args.username:
```

```
  auth=HTTPDigestAuth(args.username,args.passwd)
control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
if zoomstep == True:
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_wide_
var ' zooms;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(4)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_tele_
var ' zooms;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(4)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
```

```
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
else:
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_tele_
std';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(5)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )

payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_wide_
std';
```

```
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(5)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
```

**Command to execute with out zoom step size**

```
python3 Zoom_visca.py -ip 192.168.1.105 -username admin -passwd admin
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin', z='3', zs=False)
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":3040,"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":0,"ret":"0","status":"OK"}
```

**Command to execute with zoom step size**

```
python3 Zoom_visca.py -ip 192.168.1.105 -username admin -passwd admin -z 3 -
zs
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin', z='3', zs=True)
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":0,"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":7304,"ret":"0","status":"OK"}
```

## InsQuery

---

Querry the PTZ values for the angle.

**Parameters:**

---

***data*** Read the json table and retrieve the index and querry type. Possible querry are YawAngle, PitchAngle, RollAngle, All.

***YawAngle*** provide the yaw angle value from the PTZ.

***PitchAngle*** provide the pitch angle value from the PTZ.

***RollAngle*** provide the roll angle value from the PTZ.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## LoadUser

---

Load the active preset value from the Database and update in the state table database.

**Parameters:**

---

***enc_current_preset*** Get the preset row and check the opermation mode. The operation modes are encoder / decoder.

***cur_preset*** Provide the current presetting values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## Login

---

Set the password for the login user.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## Logout

---

Logout from the system and update in the Database.

**Parameters:**

---

***Authorization*** Get the permission level.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## Overlay

---

Add/Update the overlay values in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***rgn_idx*** Index to map overlay to.

***type*** Type of overlay to be used possible values: text or png.

***source*** In the case of text overlay this would be the source text for png overlay this is the path to the png on the encoder. Images must be uploaded to board to be used.

***location*** This is the location of the overlay. Possible values: 'top_left', 'top_right', 'top_center', 'bottom_left', 'bottom_right', 'bottom_center', 'x,y' (negative numbers not supported for x or y).

***char_size*** For text overlay this is the character size. Possible values: 16,32,64, 128.

***layer*** This will set the layer for the overly higher numbers will overlay over lower numbers if overlapping.

***alpha*** Sets the transparency of the text. Possible values: 0-255.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## OverlayStop

Stop overlay.

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***rgn_idx*** Index of overly for this channel.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzAbsoluteMove

Send absolute move command to the PTZ.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzAuxiliary

---

Send PTZ auxiliary value to the PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzClearPreset

---

Remove the Ptz preset values for the user.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzContinuousMove

---

Send continous move command to the PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzGotoPreset

---

Send preset value to PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzNewTourSpot

Get the new tour spot and update in the Database.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzPosition

Move the Ptz to the input position

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzPreset

Preset the Ptz based on the value.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzQuery

Get the ptz status and degree value from the ptz

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzRemoveTourSpot

Remove the ptz tour spot

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzSetPreset

Set the preseting value in the PTZ

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzStartTour

Start the PTZ tour

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzStop

Stop the PTZ

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## RemoveJobs

Delete the schedule jobs from the Database.

**Parameters:**

***purgelist*** List of schedule jobs to be removed.

***rowcnt*** Number of rows.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## RestartBoard

Read the operation mode. Whether its encoding / decoding. Stop the channel encoding and restart the board

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SaveCamera

Saving the camera settings in the Database.

**Parameters:**

***zoom_direct_value*** Zoom value. Possible values: 0 (wide) to 0x7ac0 (full zoom)

***white_balance_mode*** White balance mode. Default value is auto. Possible values: auto / manual / indoor/ one-push / auto trace / sodium lamp auto / sodium lamp fixed / sodium lamp outdoor.

***color_gain*** color gain. Default value is 4. Possible values: Range from 0 (60%) to 14 (200%)

***color_hue*** color hue. Default value is 7. Possible values: Range from 0 (60%) to 14 (200%)

***chroma_suppress*** Chroma suppress. Default value is 1. Possible values: 0=none, 1, 2, 3

***wb_manual_rgain_direct*** white balance manual red gain. Default value is 190. Possible values: 0 to 255

***wb_manual_bgain_direct*** white balance manual blue gain. Default value is 190. Possible values: 0 to 255

***optical_zoom_only*** optical zoom only. Default value is 0. Possible values: 0, 1

***focus_direct_value*** focus direct value. Default value is 4096. Possible values: 0x1000 to 0xf000

***manual_focus*** Manual focus. Default value is auto. Possible values: auto, manual

***exposure_mode*** Exposure mode. Default value is 0. Possible values: 0 - Auto, 10 - Shutter Priority, 11 - Iris -Priority, 3 - Manual.

***shutter*** Shutter. Default value is 7.

***iris*** Iris. Default value is 0.

***gain*** Gain. Default value is 0.

***high_sensitivity*** high sensitivity value. Default value is 3. Possible values 0, 1.

***hlc_level*** hlc level. Default value is 0. Possible values in range 0 to 3.

**hlc_level_mask** mask value of hlc level. Default value is 0. Possible values 0, 1.

***stable_zoom*** stable zoom on or off. Default value is off.

***eflip*** Default value is 5. Possible values 0, 1.

***lr_reverse*** Default value is 0. Possible values 0, 1.

***monitor_mode*** Default value is 1080p-59.94.

***genlock_source*** Default value is 2.

***zoom_step_size*** Default value is 4. Possible values in range (1 - 7).

***focus_step_size*** Default value is 1. Possible values in range (0 - 7).

***manual_icr*** Default value is Off. Possible values: On, Off, Auto, Auto Color

***img_freeze*** Default value is 0. Possible values: 0, 1.

***hr_mode*** High resolution mode. Default value is 0. Possible values: 0, 1.

***img_stabilizer*** Default value is 0. Possible values 0, 1.

***img_bw*** Image black white. Default value is 0. Possible values 0, 1.

***nr_2d_level*** Default value is 5. Possible values in range (0 - 5).

***nr_3d_level*** Default value is 4. Possible values in range (0 - 5).

***icr_threshold*** Default value is 14. Possible values in range (0-255 for 4K camera, 0-28 for HD camera).

***slow_shutter*** Default value is 0. Possible values: 0, 1.

***slow_shutter_limit*** Default value is 4. Possible values in range: (1 to 6)

***flicker_reduction*** Default value is 0. Possible values: 0, 1.

***img_stabilizer_level*** Default value is 0. Possible values: 0, 2.

***wide_dynamic_range*** Default value is 3.

***ve_brightness*** Default value is 3. Possible values in range: 0, 6.

***ve_compensation_type*** Default value is 2. Possible values in range: 0, 3.

***ve_compensation_level*** Default value is 1. Possible values in range: 0, 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SaveCameraBoson

Save color pallete settings of boson camera in the Database.

**Parameters:**

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SaveCameraDRSTamarisk

Save color pallete settings of DRS Tamarisk camera in the Database.

**Parameters:**

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SaveCameraLink

Saving camera link settings in the Database.

**Parameters:**

***pxl_format*** Pixel format. Default value is 1.

***color_table*** Color table. Default value is whitehot.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SaveCameraLx

Saving camera Lx settings in the Database.

**Parameters:**

***SONY_ExposureMode*** Exposure Mode. Default value is 32852. Possible values: 32848 (Program Auto), 32849 (Aperture Priority), 32850 (Shutter Priority), 32851 (Manual Exposure), 32852 (Intelligent Auto).

***SONY_ShutterSpeed*** Shutter speed. Defualt value is NULL.

***SONY_ISO*** ISO. Defualt value is NULL.

***SONY_ExposureComp*** Exposure comp. Defualt value is NULL.

***SONY_WhiteBalance*** White balance. Defualt value is NULL.

***SONY_ColorTemp*** Color temperature. Defualt value is NULL.

***SONY_APS_C*** APS_C. Defualt value is NULL.

***SONY_NtscPalSelect*** NTSC / PAL selection. Defualt value is NULL.

***SONY_MovieSteadyMode*** Movie steady mode. Defualt value is NULL.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveCameraTau

---

Saving camera TAU settings in the Database.

**Parameters:**

---

***color_table*** color table value .

***is_active*** Active/Deactive. Default value is 0.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveJobs

---

Save the schedule jobs in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the schedule activities.**.**

***rowcnt*** Get the number of row count.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SavePresets

---

Check the operation mode whether its in encoder or decoder. Save the preseting value of the mode in the Database.

**Parameters:**

---

***dec_current_preset*** Save the decoder current preseting values.

***enc_current_preset*** Save the encoder current preseting values.

***preset*** Settings for encoder or decoder.

***enc_channels*** Number of channels. Maximum number of channels is 2 for decoder and 4 for encoder.

***opmode*** Operation mode. Default is Encoder. Possible values: Encoder / Decoder.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveUser

---

Save the state, preset, encoder/decoder settings in the Database.

**Parameters:**

---

***enc_current_preset*** Get the encoder preseting values for the current channel.

***dec_current_preset*** Get the decoder preseting values for the current channel.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetAdv

---

Save the encoder advance settings in the Database.

**Parameters:**

---

***enc_adv_setting*** Get the encoder advance setting values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetAPConfig

---

Set the wifi and its password.

**Parameters:**

---

***passwd*** Get the wifi password

***passwden*** Get wifi enable / disable. Default value is disable.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetAudio

---

Save the audio input settings in the Database.

**Parameters:**

---

***aport*** Get the audio port. Default value is MIC. Possible values are MIC , MICL

***analog_gain_db*** Get the analog gain decibal value. Default value is 0. The value ranges are -97 to 30.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetCameraLink

---

Save the color table and pixel format values in the Database and set the color table and pixel format in the fpga.

**Parameters:**

---

***val*** Get the value for color table and pixel format.

***opt*** The options are color table and pixel format.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetConsole

---

Set the console value as ttyAMA0 or none.

**Parameters:**

---

***console_enable*** Get console value is enabled / disabled.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetDDNSEnable

Set the Dynamic Domain Name System settings in the Database.

**Parameters:**

***ddns_enable*** Get the ddns enable value. Default value is Off. Possible values; (On, Off)

***ddns_provider*** Get the ddns provider. Default value is freedns. Possible values: *(*freedns, freemyip, dyn, domains.google.com, duckdns.org, no-ip.com, tunnelbroker.net, dynv6.com, cloudxnv.net, dnspod.cn, cloudflare.com).

***ddns_username*** Provide the username.

***ddns_password*** Provide the password.

***ddns_hostname*** Hostname for DDNS login.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetDebugLevel

Set the logging level. The setting is persistent.

**Parameters:**

---

***sysdebuglevel*** Get the system debug level. Possible values 0=None, 1=Error, 2=Information, 3=Diagnostics, 4=Codeflow, 5=Dataflow.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetDevName

---

Set the device name in the Database.

**Parameters:**

---

***sysdevicename*** Get the system device name.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetDisplay

---

Controls composite output (passthru video from camera).

**Parameters:**

---

***disp_std*** Get display standard for encoder. Default value is Auto.

***disp_input*** Get display input for encoder. Default value is MACRO_DEFAULT_DISP_INPUT.

***save_only*** Possible values: (True / False)

***disp_layout*** Display layout for decoder. Default value is 1×1.

***disp_mode*** Display mode for decoder. Default value is UltraHD.

***disp_std2*** Display standard for decoder. Default value is auto.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetDSCP

---

Set the DSCP configuration settings in the Database.

**Parameters:**

---

***diff_serve*** Get the DSCP configuration value. Default value is 0x00. Possible values are ( 0x00, 0xB8, 0x88, 0x90, 0x98, 0x80, 0x68, 0x70, 0x78, 0x60, 0x48, 0x50, 0x58, 0x28, 0x30, 0x38, 0x01).

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetFpga

Set the fpga settings in the Database.

**Parameters:**

***fpgafileglob*** Get fpga value. Possible values are (boson / cvbs / tamarisk / tau2).

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetIp

Control networking settings of the Z3 device such as local IPv4 IP address, netmask, etc. Additionally control Encoder Auto-Start after boot-up and enable/disable showing of advanced WebUI settings.

**Parameters:**

***do_autostart*** Control automatic stream start after bootup. Possible values: 1 = do autostart 0 = do not autostart.

***enc_adv_setting*** Control appearance of advanced settings on WebUI. Possible values: off, on.

***ipmtu*** Control Ethernet IP Maximum Transmission Unit (MTU) size in bytes. Possible values: 576 to 1500.

***eth_speed*** Control Gigabit Ethernet auto-negotiation allowed speeds. Possible values: AUTO, AUTO-100, 1000, 100, 10.

***eth_duplex*** Control Ethernet auto-negotiation allowed duplex. Possible values: AUTO, FULL, HALF.

***local_ip*** IPv4 address.

***local_netmask*** IPv4 netmask.

***default_gw*** IPv4 default gateway.

***local_dnsip*** DNS server primary.

***local_dnsip2*** DNS server secondary.

***use_dhcp*** Controls use of DHCP or static IP. Possible values: 1 (DHCP enabled), 0 (DHCP disabled, use static IP).

***local_hostname*** Network hostname.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetLoginLimit

---

Set the login limit settings in the Database

**Parameters:**

---

***maxlogin*** mximum number of login limit. Possible value in range (3 to 99 ).

***login_window*** Login window period, the value are considered in seconds. Possible value in range (60 to 32768).

***lockout_interval*** Lockout interval values are mentioned in seconds, which is used when maximum number of login limit is reached. Possible value in range (-1 to 32768)

***login_timeout*** Login timeout

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetNFS

---

Set the NFS settings in the Database.

**Parameters:**

---

***nfs_enable*** Enable/Disable nfs

***nfs_server*** Get the nfs server ip address. Default is 192.168.1.6

***nfs_server_root*** Get the server root path / location.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetSNTP

---

Set SNTP (Simple Network Time Protocol) settings in the Database.

**Parameters:**

---

***enable*** Enable/Disable SNTP. Possible value: true, false. Default value true.

***servers*** NTP server or list of NTP servers.

***timezone*** Linux TZ database value for Timezone.

***timezone_name*** Name for a time zone. Default is America/Chicago.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetSNMP

---

Set the SNMP settings in the database.

**Parameters:**

---

***power_trap_en*** Enable/Disable power trap.

***input_loss_trap_en*** Enable/Disable input loss trap.

***input_recover_trap_en*** Enable/Disable input recover trap.

***temp_high_trap_en*** Enable/Disable temp high trap.

***temp_recover_trap_en*** Enable/Disable temperature recover trap.

***trap_hosts*** Trap hosts ipaddress. Default ip address value is 192.168.0.6.

***high_temp_suppress*** hiigh temperature. Default value is 30.

***nominal_temp_suppress*** nominal temperature default value is 30.

***snmp_v3_en*** Enable/Disable SNMP

***snmp_v3_encrypt*** SNMP v3 encrypt value.

***snmp_v3_user*** z3user.

***snmp_v3_passwd*** z3password.

***snmp_v3_usrpro*** snmp v3 usrpro default value is md5.

***snmp_v3_encrypt_passwd*** z3password.

***snmp_v3_encpro*** decrypt value.

***snmp_v3_engineid*** Default value for engineid is 0.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetNMEAEnable

---

Save the GPS NMEA enable/disable value in the Database.

**Parameters:**

---

***nmea_enable*** Enable/Disable the NMEA. Possible value: on, off.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetOnvif

---

Save the ONVIF settings in the Database.

**Parameters:**

---

***fixed_profile_max*** Get the Maximum ONVIF profiles to allow (1 or 2).

***en_persistent_audio*** Enable/Disable ONVIF persistent audio channel.

***onvif_enable*** Enable/Disable ONVIF.

***list_all_video_sources*** List all the video source.

***ptz_timeout*** Get the PTZ timeout values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetOnvifVMD

---

Save the ONVIF video motion detection and too darkness settings in the Database.

**Parameters:**

---

***vmd_enable*** Enable/Disable video motion detection.

***vmd_sens*** sensitivity value required for vmd. Possible value in range: (1 to 100).

***vmd_zone_modify*** Enable/Disable video motion and too darkness zone coordinates.

***vmd_vcrop_width*** crop width of the coordinates.

***vmd_vcrop_height*** crop height for VMD and too_darkness.

***vmd_vcrop_x*** crop startx for VMD and too_darkness.

***vmd_vcrop_y*** crop starty for VMD and too_darkness.

***td_enable*** Enable/Disable too darkness.

***td_thresh*** Threshold value for too darkness. Possible values in range: 1 to 100.

***vmd_td_channel*** Get the channel number. Possible values: 1, 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetRTSP

---

Set the RTSP settings in the Database.

**Parameters:**

---

***rtspd_port*** Get rtsp port. Default value is 554. Ports that are not in use are valid; ranging from 0 to 65535.

***rtspd_timeout*** Timeout value required for the rtsp connection. Default is 60. The range is from -1 to 32767. It is not recommended to lower than 60.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetTermSrvEnable

---

Sett the term server value in the Database.

**Parameters:**

---

***termserve_remote_enable*** Enable/Disable remote access to terminal server. Possible values are : on, off.

***termserve_shared_enable*** Enable/Disable shared access to terminal server. Possible values are : on, off.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetVideoGroup

---

Set the video group configuration settings in the Database.

**Parameters:**

---

***group*** Video config groups. Default value 0.

***nr_enable*** Enable / Disable NR. Default value is on. Possible value: on/off.

***crop_enable*** Enabled / Disable the crop.

***crop_x*** Crop x coordinates.

***crop_y*** Crop y coordinates.

***crop_width*** Crop width

***crop_height*** Crop height

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetViewport

---

Set the view port for the decoder and update in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***viewport*** viewport settings. The default value is fit.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetViVpssMode

---

Set the ViVpss mode settings in the Database.

**Parameters:**

---

***enc_vivpss_mode_enable*** Enable/Disable the vivpss mode. Default value is off. Possible values: on, off.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetWifiAP

Set the wifi network settings in the Database.

**Parameters:**

***ssid*** Get the Wifi service set identifier value.

***psk*** Get the Wifi password.

***wifi_client_local_ip*** Local IP addres.

***wifi_client_local_local_netmask*** Subnet mask for Wifi network.

***wifi_client_default_gw*** Default Gateway be used for Wifi Network.

***wifi_client_use_dhcp*** DHCP server used with Wifi network for DNS resolutions.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetZFinderEnable

Save ZFinder enable / disable in the Databse.

**Parameters:**

---

***zfinder_enable*** Get ZFinder enable/disable value. Possible values: off, on

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StartChannel

---

Start the encoder channel and update the state as running in the Database. Once you start the channel, it will not transmit data until the video input is detected.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***actv_preset***

***vsource*** video source. Default value is Camera1. Possible values: Camera1 / Camer2 / HDMI1.

***vres*** video resolution. Default value is Follow Input. Possible values: Follow Input, 3840 x 2160, 1920 x 1080, 1440 x 1080, 1280 x 1024, 1280 x 720, 1024 x 768, 960 x 720, 1024 x 576.

***vcodec*** video codec. Default value is H265. Possible values: H265, H264, MJPEG, none.

***vgdr*** video gradual data refresh. Default value is on. Possible value on, off.

***vprofile*** video profile. Default value is high. Possible value high, main, baseline.

***vratectrl*** video rate control. Default value is cbr. Possible value cbr, vbr.

**vbitrate** video bit rate.

***vframeratediv*** video frame rate. Default value is 1. Possible values: 1 (Full), 2 (Half), 4 (Quarter), 6 (Sixth).

***vgopsize*** video group of picture size. Default value is 60. Possible values: 1 (1 Frame Only), 5 (5 Frames), 8 (8 Frames), 10 (10 Frames), 12 (12 Frames), 15 (15 Frames), 25 (25 Frames), 30 (30 Frames), 50 (50 Frames), 60 (60 Frames), 90 (90 Frames), 100 (100 Frames), 120 (120 Frames), 200 (200 Frames), 240 (240 Frames).

***vprotocol*** video protocol.

***vdest*** video destination port.

***vpid*** video packet identifier. Default value is 221.

***vdelay*** video delay. Default value is 300.

***pcrpid*** PCR packet identifier. Default value is 521.

***pcrinterval*** Default value is 50.

***pmtpid*** program map table packet identifier. Default value is 31.

***tsrate*** bit rate for transport stream. Default value is 5000K.

***tslowlat*** transport stream low latency mode. Default value is off. Possible values: on, off.

***feconoff*** Forward error correction in transport stream enable/ disable. Default value is off. Possible values: on, off.

***fecrow*** Forward error correction row. Default value is 1.

***feccol*** Forward error correction column, Default value is 5.

***zixioverhead*** ZIXI Forward Error Correction total overhead as a percentage of transport rate. Default value is 15.

***zixilatency*** ZIXI target latency (to allow ARQ) in milliseconds. Default value is 500.

***zixifecblock*** ZIXI Forward Error Correction block size in milliseconds. Max value is 1/2 of Zixi Latency. Default value is 50.

***zixirateadjen*** ZIXI dynamic bitrate adjustment according to network conditions. Default value is on. Possible values: on, off.

***zixiauthen*** ZIXI dynamic authentication enable/disable . Default value is off. Possible values: on, off.

***zixisession*** ZIXI session id. Default value is test.

***zixiuser*** ZIXI user identifier. Default value is user.

***aenable*** Audio enable/disable. Default value is on. Possible values: on, off.asource. Default value is MICL.

***apair*** Stereo pair select for digital inputs. Default value is 0. Possible values: (0 (1 2), 1 (3 4), 2 (5 6), 3 (7 8), 260 (1 2,3 4), 548 (1 2,3 4,5 6), 1252 (1 2,3 4,5 6,7 8)).

***acodec*** Audio codec. Default value is fdk_aaclc.

***abitrate*** Audio bit rate. Default value is 128000.

***asamplerate*** Audio sample rate. Default value is 48000.

***aport*** Audio port. Default value is 8700.

***apid*** Audio packet identifier. Default value is 120.

***aptspcr*** Maximum difference between the PTS and PCR in the audio stream Default value is 250.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StartMTS

---

Start the MTS.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StopChannel

---

Stop the encoder channel and update the state in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StopMTS

---

Stop the MTS.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## TempStatus

---

Get the current temperature status for CPU and lense.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## update_require_web_login

Update/Change whether web login credential required or not in the Database.

**Parameters:**

***require*** Required web login value. Possible values: 1 or 0

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## UpdatePtz

Update the PTZ value in the Database.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the PTZ values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## UpdatePtzPreset

---

Update the PTZ preset value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ preset values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## UpdatePtzTourSpots

---

Update the PTZ tour spots value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ tour spots values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## UpdateTerm

---

Control serial port terminal server for VISCA

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the terminal server values.

***term_mode*** Possible values: client, server

***term_protocol*** Possible values: clearchannel, telnet

***term_localport*** Local TCP port (for server mode)

***term_servaddr*** Remote IP address (for client mode only)

***term_servport*** Remote TCP port (for client mode only)

***term_data_bits*** number of data bits.

***term_parity*** Possible values: None, Odd, Even, Mark, Space

***term_stop_bits*** Stop bits sent at the end of every character.

***term_baudrate*** Possible values: 9600, 19200, 38400, 57600, 115200

***term_devicefile*** device fine name

***term_function*** functionality

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## user_add

---

Add a new user in the Database.

**Parameters:**

---

***Name*** Get new user name

***Level*** Get the level of permission for the new user name. Possible value: 0 to 7.

***Password*** Get the password for the new user name.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## user_remove

---

Remove the user from the database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the user name.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## user_update_level

Update/change the level of permission to the given user name.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the user name and level of the permission.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## user_update_password

Update/change the password to the given user name.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the user name and password.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

### download_remove

```
http://Encoder_IP/cgi-bin/control.cgi?action=download_remove&data=FILENAME

http://192.168.0.120/cgi-bin/control.cgi?action=download_remove&data=/media/sda/MOV1_000040.mp4
```

remove the downloaded media content from the usb drive or sd card.

**Parameters:**

***data*** Filename from the downloaded media content to be removed from the removable media device.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## Generic Control (GET)

### 8021x

Get the exthernet 802x information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***interface*** TEXT,

***enable*** TEXT.

***eap*** TEXT.

***anonymous_identity*** TEXT.

***identity*** TEXT.

***password*** TEXT.

***private_key_password*** TEXT.

---

## GetCronJobs

---

Get the schedule job from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***enable*** INTEGER,

***cron*** TEXT.

***func*** TEXT.

***func_args*** TEXT.

***type*** TEXT.

***minutes*** INTEGER.

***hours*** INTEGER.

***day*** INTEGER.

***month*** INTEGER.

***cron*** TEXT.

---

## GetNFSMount

---

Get the Network File Sharing mounted infromation from the device.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mounted*** TEXT,

---

## aspect_info

---

Get the video resolution for the current channel from the Database and calculate the aspect ratio.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***numerator*** INTEGER,

***denominator*** INTEGER.

---

## boardinfo

---

Get the board and model information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***logo_enable*** TEXT,

***logo_filename*** TEXT.

***logo_width*** INTEGER.

***logo_height*** INTEGER.

***logo_blob*** BLOB.

***ico_blob*** BLOB.

***model_enable*** TEXT.

***model_name*** TEXT.

---

## cam_state

Get the current camera state information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT,

***enc_channels*** TEXT.

***opmode*** TEXT.

***enc_current_preset*** TEXT.

***dec_current_preset*** TEXT.

## camera

Get the current camera settings from the Database.

**Parameters:**

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***zoom_direct_value*** INTEGER.

***white_balance_mode*** TEXT.

***color_gain*** INTEGER.

***color_hue*** INTEGER.

***chroma_suppress*** INTEGER.

***wb_manual_rgain_direct*** INTEGER.

***wb_manual_bgain_direct*** INTEGER.

***visca_localport*** TEXT.

***optical_zoom_only*** BOOL. 0 or 1.

***focus_direct_value*** INTEGER.

***manual_focus*** TEXT.

***flexio_localport*** TEXT.

***exposure_mode*** INTEGER.

***shutter*** INTEGER.

***iris*** INTEGER.

***gain*** INTEGER.

***high_sensitivity*** INTEGER.

***hlc_level*** INTEGER.

***hlc_level_mask*** INTEGER.

***stable_zoom*** INTEGER.

***eflip*** INTEGER.

***lr_reverse*** INTEGER.

***monitor_mode*** TEXT.

***genlock_source*** INTEGER.

***manual_icr*** TEXT.

***zoom_step_size*** INTEGER.

***focus_step_size*** INTEGER.

***img_freeze*** INTEGER.

***hr_mode*** INTEGER.

***img_stabilizer*** INTEGER.

***img_bw*** INTEGER.

***nr_2d_level*** INTEGER.

***nr_3d_level*** INTEGER.

***icr_threshold*** INTEGER.

***slow_shutter*** BOOL. 0 or 1.

***slow_shutter_limit*** INTEGER.

***flicker_reduction*** BOOL. 0 or 1.

***img_stabilizer_level*** INTEGER.

***wide_dynamic_range*** TEXT.

***ve_brightness*** TEXT.

***ve_compensation_type*** TEXT.

***ve_compensation_level*** TEXT.

***tab_index*** TEXT.

***imgvflip*** INTEGER.

***imghflip*** INTEGER.

***imgrotate*** INTEGER.

---

## camera_exposure

---

Get the current camera exposure values from the visca camera.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***ae_mode*** TEXT,

***shutter*** TEXT.

***iris*** TEXT.

***gain_inquiry*** TEXT.

***sensitivity*** TEXT.

***hls_inquiry*** TEXT.

***min_shutter*** INTEGER.

***max_shutter*** INTEGER.

***shutter_label*** TEXT.

***min_iris*** INTEGER.

***max_iris*** INTEGER.

***iris_label*** TEXT.

***min_gain*** INTEGER.

***max_gain*** INTEGER.

***gain_label*** TEXT.

---

## camera_monitor_mode

---

Get the current camera monitor mode values from the visca camera.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***modes*** TEXT.

***current_mode*** TEXT.

---

## dec

---

Get the current decoder settings from the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT.

***channel*** INTEGER.

***url*** TEXT.

***aenable*** TEXT.

***viewport*** TEXT.

***latency_mode*** TEXT.

***max_latency_ms*** TEXT.

***srt_mode*** TEXT.

***srt_latency*** TEXT.

***srt_decrypt*** TEXT.

***srt_pass*** TEXT.

***zixi_fec*** TEXT.

***zixi_fecoverhead*** TEXT.

***zixi_fecblock*** TEXT.

***zixi_latency*** TEXT.

***zixi_decrypt*** TEXT.

***zixi_pass*** TEXT.

***rtsp_flags*** TEXT.

---

## download

---

Get the downloaded file information from the removable storage media.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***job*** Optional***.*** The job from the schedule tab.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***FileName*** TEXT.

***FileSize*** TEXT.

***LastModified*** TEXT.

---

## drs_tamarisk_settings

Get the camera model of drs tamarisk settings fromt the Database.

**Parameters:**

***job*** Optional***.*** The job from the schedule tab.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***color_pallette*** TEXT.

***tab_index*** TEXT.

***color_pallette_disabled*** TEXT.

## enc

Get the current channel encoder settings from the Database.

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT,

***channel*** INTEGER.

***vbitrate*** TEXT.

***vframeratediv*** INTEGER.

***vsource*** TEXT.

***vcodec*** TEXT.

***vprofile*** TEXT.

***vprotocol*** TEXT.

***avmux_index*** TEXT.

***vgdr*** TEXT.

***vdest*** TEXT.

***aenable*** TEXT.

***acodec*** TEXT.

***abitrate*** TEXT.

***asamplerate*** TEXT.

***aport*** TEXT.

***apair*** INTEGER.

***vres*** TEXT.

***vgopsize*** INTEGER.

***vpid*** INTEGER.

***apid*** INTEGER.

***zixiauthen*** TEXT.

***zixisession*** TEXT.

***zixiuser*** TEXT.

***zixioverhead*** TEXT.

***zixifecblock*** TEXT.

***zixilatency*** TEXT.

***zixirateadjen*** TEXT.

***fecrow*** INTEGER.

***feccol*** INTEGER.

***feconoff*** TEXT.

***aptspcr*** INTEGER.

***tslowlat*** TEXT.

***tsrate*** TEXT.

***pcrpid*** INTEGER.

***pcrinterval*** INTEGER.

***pmtpid*** INTEGER.

***klvenable*** TEXT.

***klvmode*** TEXT.

***klvmuxmethod*** TEXT.

***klvsrc*** TEXT.

***klvbrate*** TEXT.

***klvserialbaud*** TEXT.

***klvpid*** TEXT.

***vratectrl*** TEXT.

***vdelay*** INTEGER.

***storage*** TEXT.

***fprefix*** TEXT.

***asource*** TEXT.

***authonoff*** TEXT.

***auth_user*** TEXT.

***auth_passwd*** TEXT.

***auxonoff*** TEXT.

***filesize*** TEXT.

***nfstrength*** TEXT.

***telopenable*** TEXT.

***teloptext*** TEXT.

***teloplocation*** TEXT.

***telopcharsize*** TEXT.

***teloptextcolor*** TEXT.

***telopoutlineenable*** TEXT.

***telopoutlinecolor*** TEXT.

***gps_overlay_enable*** TEXT.

***gps_overlay_device*** TEXT.

***gps_overlay_location*** TEXT.

***gps_overlay_char_size*** TEXT.

***pipenable*** TEXT.

***piplocation*** TEXT.

***vquality*** TEXT.

***vinterlacemode*** TEXT.

***vmulticastdest*** TEXT.

***amulticastdest*** TEXT.

***rtsp_auth_enable*** TEXT.

***rtsp_auth_username*** TEXT.

***rtsp_auth_password*** TEXT.

***rtsp_transport_mode*** TEXT.

***lowdelay_opt*** TEXT.

***vcropaspect*** TEXT.

***vcrop_enable*** TEXT.

***vcrop_width*** INTEGER.

***vcrop_height*** INTEGER.

***vcrop_x*** INTEGER.

***vcrop_y*** INTEGER.

***rotate_enable*** INTEGER.

***rotate_angle*** INTEGER.

***rtmp265_enable*** TEXT.

***srt_pass*** TEXT.

***srt_encrypt*** INTEGER.

***srt_mode*** INTEGER.

***srt_destAddr*** TEXT.

***mmulticastdest*** TEXT.

***mport*** TEXT.

***frame_loss_mode*** TEXT.

***frame_loss_gap*** INTEGER.

***frame_loss_overshot*** INTEGER.

***mounts*** TEXT.

***source_status_str*** TEXT.

---

## filepicker

---

Get the file type and its information from the removable storage media.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***filepicker_filename*** TEXT,

***filepicker_size*** INTEGER.

***filepicker_lastmodified*** TEXT.

---

## fmt

---

Get the removable storage infromation from the device using fdisk command.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mounts*** TEXT,

---

## focusStepSize

---

Get the current camera focus step values from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***focus_step_size*** INTEGER,

## fpga

Get the current fpga from firmware filename.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***fpgafileglob*** TEXT,

***options*** TEXT.

## history

Get the decoder history infromation from the Database.

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***channel*** TEXT,

***histidx*** TEXT.

***url*** TEXT.

---

## ipinfo

---

Get the internet protocol infromation from the board.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***local_ip*** TEXT,

***local_netmask*** TEXT.

***default_gw*** TEXT.

***ipmtu*** TEXT.

***eth_speed*** TEXT.

***eth_duplex*** TEXT.

***do_autostart*** BOOLEAN.

---

## onvif

---

Get the ONVIF miscallenous settings from the Database.

**Parameters:**

---

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***list_all_video_sources*** INTEGER,

***dns_from_dhcp*** INTEGER,

***ntp_from_dhcp*** INTEGER,

***ntp_enable*** INTEGER,

***daylightsaving*** INTEGER,

***timezone*** TEXT.

***http_enable*** INTEGER,

***http_port*** INTEGER,

***https_enable*** INTEGER,

***https_port*** INTEGER,

***rtsp_enable*** INTEGER,

***rtsp_port*** INTEGER,

***discoverable*** INTEGER,

***ntp*** TEXT.

***dns_search*** TEXT.

***ptz_timeout*** TEXT.

***primary fixed_profile_max*** INTEGER,

***secondary fixed_profile_max*** INTEGER,

***en_persistent_audio*** TEXT.

***onvif_enable*** BOOLEAN.

***vmd_enable*** BOOLEAN.

## onvif_vmd

Get the ONVIF video motion detection and too darkness settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***vmd_sens*** INTEGER,

***vmd_zone_modify*** TEXT.

***vmd_vcrop_width*** INTEGER.

***vmd_vcrop_height*** INTEGER.

***vmd_vcrop_x*** INTEGER.

***vmd_vcrop_y*** INTEGER.

***td_enable*** TEXT.

***td_thresh*** INTEGER.

***vmd_td_channel*** TEXT.

---

## permission

---

Get the current permission settings for admin, operators, and users from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***username*** TEXT,

***userlevel*** TEXT.

***api_command*** TEXT.

***superadmin*** TEXT.

**admin** TEXT.

***operator1*** TEXT.

***operator2*** TEXT.

***user1*** TEXT.

***user2*** TEXT.

***user3*** TEXT.

***anonymous*** TEXT.

---

## ptz

---

Get the current PTZ settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***token*** TEXT,

***node*** TEXT.

***name*** TEXT.

***enabled*** BOOLEAN.

***pelcoaddr*** INTEGER.

***tcpport*** INTEGER.

***ptztype*** TEXT.

***flip*** TEXT.

***pos*** TEXT.

***maxtilt*** INTEGER.

---

## ptz_preset

---

Get the PTZ preseting values from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***profiletoken*** TEXT,

***token*** TEXT.

***name*** TEXT.

***pan*** FLOAT.

***tilt*** FLOAT.

***zoom*** FLOAT.

---

## ptz_tour

---

Get the PTZ tour information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***ProfileToken*** TEXT,

***PresetTourToken*** TEXT.

***name*** TEXT.

***autostart*** BOOLEAN.

***RecurringTime*** INTEGER.

***RecurringDuration*** TEXT.

***PresetTourDirection*** INTEGER.

***RandomPresetOrder*** BOOLEAN.

***ProfileToken*** TEXT.

***PresetTourToken*** TEXT.

***RowIndex*** INTEGER.

***PresetDetailToken*** TEXT.

***PresetDetailHome*** BOOLEAN.

***StayTime*** TEXT.

***pan*** FLOAT.

***tilt*** FLOAT.

***zoom*** FLOAT.

***pspeed*** FLOAT.

***tspeed*** FLOAT.

***zspeed*** FLOAT.

## snmp

---

Get the Simple Network Management Protocol values from the Database

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***power_trap_en*** INTEGER,

***input_loss_trap_en*** INTEGER.

***input_recover_trap_en*** INTEGER.

***temp_high_trap_en*** INTEGER.

***temp_recover_trap_en*** INTEGER.

***high_temp*** INTEGER.

***nominal_temp*** INTEGER.

***high_temp_suppress*** INTEGER.

***nominal_temp_suppress*** INTEGER.

***trap_hosts*** INTEGER.

***snmp_v3_en*** INTEGER.

***snmp_v3_user*** TEXT.

***snmp_v3_passwd*** TEXT.

***snmp_v3_usrpro*** TEXT.

***snmp_v3_encrypt*** INTEGER.

***snmp_v3_encrypt_passwd*** TEXT.

***snmp_v3_encpro*** TEXT.

***snmp_v3_engineid*** TEXT.

## ssl

Get the Secure Socket Layer settings from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***https_enable*** TEXT,

***http_enable*** TEXT.

***https_port*** INTEGER.

***http_port*** INTEGER.

***cert_status*** TEXT.

***key_status*** TEXT.

## stats

Get the available memory infromation from the device.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mem_free_kb*** TEXT,

---

## sys

---

Get the current system settings from the Database and the ip information from the device.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***local_ip*** TEXT,

***local_netmask*** TEXT.

***preset*** TEXT.

***enc_channels*** TEXT.

***opmode*** TEXT.

***username*** TEXT.

***userlevel*** TEXT.

***do_autostart*** BOOLEAN.

***syspassword*** TEXT.

***session_id*** TEXT.

***disp_std*** TEXT.

***disp_std2*** TEXT.

***disp_input*** TEXT.

***disp_mode*** TEXT.

***disp_layout*** TEXT.

***enc_adv_setting*** TEXT.

***enc_vivpss_mode_enable*** TEXT.

***enable_sntp*** TEXT.

***sntp_servers*** TEXT.

***enable_snmp*** TEXT.

***sysdevicename*** TEXT.

***timezone*** TEXT.

***timezone_name*** TEXT.

***termserve_remote_enable*** TEXT.

***termserve_shared_enable*** TEXT.

***zfinder_enable*** TEXT.

***diff_serve*** INTEGER.

***enable_ptp*** TEXT.

***ptp_role*** TEXT.

***nmea_enable*** TEXT.

***nfs_enable*** TEXT.

***nfs_server*** TEXT.

***nfs_server_root*** TEXT.

***ddns_enable*** TEXT.

***ddns_provider*** TEXT.

***ddns_username*** TEXT.

***ddns_password*** TEXT.

***rtspd_port*** INTEGER.

***rtspd_timeout*** INTEGER.

***maxlogin*** INTEGER.

***login_window*** INTEGER.

***lockout_interval*** INTEGER.

***login_timeout*** INTEGER.

***require_web_login*** BOOLEAN.

## term

Get the remote terminal information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***term_mode*** TEXT,

***term_protocol*** TEXT.

***term_localport*** TEXT.

***term_servaddr*** TEXT.

***term_servport*** TEXT.

***term_data_bits*** TEXT.

***term_parity*** TEXT.

***term_stop_bits*** TEXT.

***term_baudrate*** TEXT.

***term_devicefile*** TEXT.

***term_function*** TEXT.

## users

Get the ONVIF user information and the permission level from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***Name*** TEXT,

***Level*** INTEGER.

***Password*** TEXT.

***Password_MD5*** TEXT.

***Password_SHA256*** TEXT.

***Password_SHA512_256*** TEXT.

---

## video_group

---

Get the current video group settings from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***vgroup*** INTEGER.

***crop_enable*** TEXT.

***crop_width*** INTEGER.

***crop_height*** INTEGER.

***crop_x*** INTEGER.

***crop_y*** INTEGER.

***nr_enable*** TEXT.

---

## wifi

---

Get the Wifi details from the device and wifi client information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***networks*** TEXT,

***cur_network*** TEXT,

***cur_pass*** TEXT,

***ap_pass*** TEXT,

***ap_pass_en*** BOOLEAN.

***ap_ssid*** TEXT,

***wifi_client_local_ip*** TEXT,

***wifi_client_local_netmask*** TEXT.

***wifi_client_default_gw*** TEXT.

***wifi_client_use_dhcp*** BOOLEAN.

---

### zoomStepSize

---

Get the current camera zoom step values from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***zoom_step_size*** INTEGER,

---

# File Upload for MFR or UPD Image Control

## Upload Image (POST)

---

Upload the z3 image for e.g. (z3-30-0139-mfr.img or z3-30-0139-upd.img) in the z3 device.

Before uploading the new image, it checks, whether the device has enough free memory or not and removes the coredump files. If the image is *mfr.img then it removes the default database from the data partition. If its *upd.img then it will not remove the database values.

Note:
It takes approximately 5 minutes to update; DO NOT SEND ANY OTHER API Commands after uploading the image for at least 5 minutes. (time.sleep(300) or equivalent in scripts)

**Parameters:**

---

***img_file*** Get the file name which needs to be flashed in the z3 device.

**Returns:**

---

Flash the image and reboot the device.

Note:
It takes approximately 5 minutes to update; DO NOT SEND ANY OTHER API Commands after uploading the image for at least 5 minutes. (time.sleep(300) or equivalent in scripts)

---

# Download File and Remove File Control

See **download_remove** to remove files from the **Generic Actions (POST)** action

## Download Media (GET)

```
http://<ENCODER_IP>/download_media.cgi?mediafile=/location/filename
```

For e.g.

```
http://<ENCODER_IP>/download_media.cgi?mediafile=/media/sda1/MOV1_000001.mp4
```

---

Download the media file. If the output format is mp4 or TS file, then only z3 device allows downloading the files. Also provides the option for deleting the file.

**Parameters:**

---

***mediafile*** Get the media file name.

**Returns:**

---

JSON formatted table with values below.

ret Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***File Name*** Name of the download file name .

***File Size*** Size of the download file name.

***Last Modified*** Last modified the file date and time.

---

# download.html to get list of downloadable files

```
http://<ENCODER_IP>/download.html?chn=1
```

chn **=[1,2,3,4]**

List the download content. The content will be either .mp4 or .ts file format.

# Encoder Actions (POST)

## SetEncoder

Write encoder settings to active preset. See Python example code for Seting Configuration below

## EncoderStatus

Requests an update to encoder_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

## StreamStatus

Requests an update to stream_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

## AStreamStatus

Requests an update to astream_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

## SourceStatus

Requests an update to source_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

# Setting Encoder Configuration

To set the encode configuration per channel the **SetEncoder** action is used with a post. All variables should be present or they will be replaced with a defualt that may not be valid for your application. More than one channel can be set per command. It is recommended you first read the current settings from the encoder then modify them and send the new settings back as in the Python example below:

```
import requests, json, sys

server_url='http://192.168.0.120/cgi-bin/control.cgi'
channel = 1

enc_cfg = requests.get(server_url, params='ctrl=enc&chn={}'.format(channel))
print enc_cfg.json()

enc_cfg_json=enc_cfg.json()
enc_cfg_json['vframeratediv'] = 2 #set frame rate divider to 2
new_cfg = {}
new_cfg['action'] = 'SetEncoder'
for key, value in enc_cfg_json.iteritems():
print "key = {} value = {}".format(key,value)
new_idx = "enc_{}_{}".format(channel,key)
print "setting {} to {}".format(new_idx,value)
new_cfg[new_idx] = value

print new_cfg
requests.post(server_url, data=new_cfg)
```

# snapshot.cgi Control

```
http://<ENCODER_IP>/snapshot.cgi?size=3840x2160&quality=100

http://<ENCODER_IP>/snapshot.cgi?chn=1&size=1280x720&quality=80
```

valid options are:

* *size **=[any supported valid resolution]**

chn **=[1,2,3,4]**

```
quality** =[0-100] supported resolutions are: 4K Models Only 3840×2160
2560×1440 4K and HD Models 1920×1080
1440×1080
1280×1024
1280×720
```

```
1024×768
960×720
1024×576
800×600
720×576
704×576
720×480
640×512
640×480
640×360
352×576
420×380
352×288
352×240
336×256
320×240
320×180 **
```

# Reading Statistics

**Python example of updating and reading encoder statistics:**

```
import requests, json, sys
server_url='http://192.168.0.120/cgi-bin/control.cgi'

requests.post(server_url, {'action': 'EncoderStatus'} )
requests.post(server_url, {'action': 'StreamStatus'}  )
requests.post(server_url, {'action': 'AStreamStatus'} )
requests.post(server_url, {'action': 'SourceStatus'}  )
requests.post(server_url, {'action': 'TempStatus'}  )

print requests.get(server_url, params='ctrl=stats&chn=null').json()
```

**Example output:**

```
{u'status': u'OK',
u'astream_status_str': u'\tChannel 6 Codec fdk_aaclc Samplerate 48000 Input
MICL Frames 32287820  OK',
u'temp_status_str': u' LENS 29  CPU 78.700  FPGA 89.8  OK',
u'ret': u'0',
u'encoder_status_str': u'   *** Encode Bitstream Received Statistics ***
CH | Bitrate (Kbps) | Actual Bitrate | FPS  | Actual FPS | Key-frame FPS |
Width | Height   ------------------------------------------------------------
-------------------------------------------------
1 |        4000.00 |        4092.18 | 60.0 |     59.8 |           1.0 |
1920 |   1080  |  OK',
u'source_status_str': u' CAMERA 1920x1080p 60.00 fps\n',
u'stream_status_str': u'Channel 1 URL rtsp Frames 12883581  OK'}
```

## Authentication

By default, no authentication is required to access the HTTP API.

To enable authentication, go to the “System” tab “Device Management” section. Click on the “Set Password” button.

The username will be “admin” The authentication method is HTTP Digest authentication. HTTP Basic authentication is not supported.

#### Python example code of HTTP Digest authentication

```
import requests,json, sys
from requests.auth import HTTPDigestAuth

auth=HTTPDigestAuth('admin','mypassword')

print requests.get(server_url, params='ctrl=stats&chn=null',
auth=auth).json()
```

# Appendix A: Sample Python Scripts

The sample python scripts below require Python3.x.

The requests package must be installed.

You can install using

1) Use the Ubuntu package manager

sudo apt install python3-requests

2) Use the Python3 native package installer

sudo pip3 install requests

## SaveUser (saveUser_DIGEST_ARGPARSE.py)

Save the state, preset, encoder/decoder settings for the user.

#### Python example code of HTTP Digest SaveUser

```
import requests
import json
```

```
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
```

```
payload = "action=SaveUser&enc_current_preset=charles";
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   setip_respose response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -
passwd admin
```

**Example output:**

```
SaveUser response: {"ret":"0","status":"OK"}
```

# SetIp (set_ip_DIGEST_ARGPARSE.py)

Control networking settings of the Z3 device such as local IPv4 IP address, netmask, DNS Server primary, DNS Server secondary, etc. Additionally control Encoder Auto-Start after boot-up and enable/disable showing of advanced WebUI settings.

**Python example code of HTTP Digest SetIp**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
```

```
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-newip', help='Set this IP')
parser.add_argument('-mask', help='Subnet Mask',default='255.255.255.0')
parser.add_argument('-gateway', help='Network Gateway',default='172.30.1.1')
parser.add_argument('-dns1', help='DNS 1',default='8.8.8.8')
parser.add_argument('-dns2', help='DNS 2',default='8.8.4.4')
parser.add_argument('-ipmtu', help='MTU Speed',default=1500)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
payload =
"action=SetIp&local_ip=172.30.1.112&local_netmask=255.255.255.0&default_gw=1
72.30.1.1&local_dnsip=8.8.8.8&local_dnsip2=8.8.4.4&use_dhcp=no&ipmtu=1500ð_s
peed=AUTOð_duplex=AUTO&do_autostart=1&enc_adv_setting=off";
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   setip_respose response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 set_ip_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -passwd
admin
```

**Example output:**

```
dns1='8.8.8.8', dns2='8.8.4.4', gateway='172.30.1.1', ip='192.168.1.105',
ipmtu=1500, mask='255.255.255.0', newip=None, passwd='admin',
username='admin'
setip_respose response: {"ret":"0","status":"OK"}
```

# cam_state (sys_DIGEST_ARGPARSE.py)

Get the current camera state information from the Database.

**Python example code of HTTP Digest camera state**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
```

```
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}

payload='ctrl=cam_state&chn=null';
rs = http_request_get(payload)
print_immediately(
  "{:>6}: {}   Sys response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
```

```
)
```

**Command to execute**

```
python3 sys_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -passwd
admin
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin')
Sys response:
{"camera1_if_type":"Sony_LVDS","camera2_if_type":"CVBS","enc_channels":"1,2"
,"enc_current_preset":"charles","fpgafileglob":"fpga_none.bin|fpga2_0139_cvb
s.bin","preview_auto_start":"1","ret":"0","visca_present":"true"}
```

# CameraControl for Boson camera (Zoom_Boson.py)

Send control commands to boson camera. Only one command can be sent at a time;

**Python example code of HTTP Digest Camera control for Boson camera**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
import sys
from requests.auth import HTTPDigestAuth

if sys.version_info[0] <3:
  raise Exception("Python 3 or a more recent version is required.")

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    # print(msg, flush=True)
    print(msg)
    sys.stdout.flush()

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
```

```
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)
if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address
# Set Zoom STEP
zooms=args.zs;
zoomstep=args.zoom_step;

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)
control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
```

```
# Boson min Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000000000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson mid Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000015000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson mid Zoom xB0 yA0
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000015000000B0000000A0';
rs = http_request(payload)
print_immediately(
```

```
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson max Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000030000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson no Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000000000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
```

```
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 Zoom_Boson.py -ip 172.28.30.103
```

**Example output:**

```
(ip='172.28.30.103', passwd='admin', username=None, zoom_step=False, zs='3')
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000000000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000015000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000015000000B0000000A0","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000030000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
```

# CameraControl for Sony Visca camera (Zoom_visca.py)

Send control commands to visca camera. Only one command can be sent at a time;

**Python example code of HTTP Digest Camera control for Visca camera**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
import sys
from requests.auth import HTTPDigestAuth
```

```
if sys.version_info[0] <3:
  raise Exception("Python 3 or a more recent version is required.")

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    # print(msg, flush=True)
    print(msg)
    sys.stdout.flush()

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
```

```
required',default='admin')
parser.add_argument('-z', help='Zoom STEP level 0 - 7; Default
3',default='3')
parser.add_argument('-zs', help='Do Zoom STEP defined using -z, else normal
zoom',action='store_true')
args = parser.parse_args()
print(args)
if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address
# Set Zoom STEP
zooms=args.z;
zoomstep=args.zs;

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)
control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
if zoomstep == True:
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_wide_
var ' zooms;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(4)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
```

```
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_tele_
var ' zooms;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(4)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
else:
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_tele_
std';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(5)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
```

```
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )

payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_wide_
std';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(5)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
```

**Command to execute with out zoom step size**

```
python3 Zoom_visca.py -ip 192.168.1.105 -username admin -passwd admin
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin', z='3', zs=False)
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":3040,"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":0,"ret":"0","status":"OK"}
```

**Command to execute with zoom step size**

```
python3 Zoom_visca.py -ip 192.168.1.105 -username admin -passwd admin -z 3 -
zs
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin', z='3', zs=True)
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":0,"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":7304,"ret":"0","status":"OK"}
```

## SetRTSP

---

Set the RTSP settings in the Database.

**Parameters:**

---

***rtspd_port*** Get rtsp port. Default value is 554.

***rtspd_timeout*** Timeout value required for the rtsp connection.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetRTSP response: {"ret":"0","status":"OK"}
```

---

## SetTermSrvEnable

---

Set the term server value in the Database.

***termserve_remote_enable*** on:
Allows for remote connections outside of the unit, via eth0 or wlan0, to the terminal server.

***termserve_remote_enable*** off:
Only allow local loopback access.

***termserve_shared_enable*** on:
Allows multiple connections at the same time but only the last one that has access will get the response.

***termserve_shared_enable*** off:
Only one connection at a time so you are sure of the response. If you have your app, it should close the connection every time it gets a response so the other processes that need access will still work.

#### Parameters:

---

***termserve_remote_enable*** Enable/Disable remote access to terminal server. Possible values are : on, off.

***termserve_shared_enable*** Enable/Disable shared access to terminal server. Possible values are : on, off.

#### Returns:

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetTermSrvEnable response: {"ret":"0","status":"OK"}
```

---

## SetVideoGroup

---

Set the video group configuration settings in the Database.

**Parameters:**

---

***group*** Video config groups. Default value 0.

***nr_enable*** Enable / Disable NR. Default value is on. Possible value: on/off.

***crop_enable*** Enabled / Disable the crop.

***crop_x*** Crop x coordinates.

***crop_y*** Crop y coordinates.

***crop_width*** Crop width

***crop_height*** Crop height

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
SetVideoGroup response: {"ret":"0","status":"OK"}
```

---

## SetViewport

---

Set the view port for the decoder and update in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***viewport*** viewport settings. The default value is fit.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetViewport response: {"ret":"0","status":"OK"}
```

## SetViVpssMode

Set the ViVpss mode settings in the Database.

**Parameters:**

***enc_vivpss_mode_enable*** Enable/Disable the vivpss mode. Default value is off. Possible values: on, off.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetViVpssMode response: {"ret":"0","status":"OK"}
```

## SetWifiAP

---

Set the wifi network settings in the Database.

**Parameters:**

---

***ssid*** Get the Wifi service set identifier value.

***psk*** Get the Wifi password.

***wifi_client_local_ip*** Local IP addres.

***wifi_client_local_local_netmask*** Subnet mask for Wifi network.

***wifi_client_default_gw*** Default Gateway be used for Wifi Network.

***wifi_client_use_dhcp*** DHCP server used with Wifi network for DNS resolutions.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetZFinderEnable

---

Save ZFinder enable / disable in the Databse.

**Parameters:**

---

***zfinder_enable*** Get ZFinder enable/disable value. Possible values: off, on

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
SetZFinderEnable response: {"ret":"0","status":"OK"}
```

## StartChannel

Start the encoder channel and update the state as running in the Database. Once you start the channel, it will not transmit data until the video input is detected.

**Parameters:**

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***actv_preset***

***vsource*** video source. Default value is Camera1. Possible values: Camera1 / Camer2 / HDMI1.

***vres*** video resolution. Default value is Follow Input. Possible values: Follow Input, 3840 x 2160, 1920 x 1080, 1440 x 1080, 1280 x 1024, 1280 x 720, 1024 x 768, 960 x 720, 1024 x 576.

***vcodec*** video codec. Default value is H265. Possible values: H265, H264, MJPEG, none.

***vgdr*** video gradual data refresh. Default value is on. Possible value on, off.

***vprofile*** video profile. Default value is high. Possible value high, main, baseline.

***vratectrl*** video rate control. Default value is cbr. Possible value cbr, vbr.

**vbitrate** video bit rate.

***vframeratediv*** video frame rate. Default value is 1. Possible values: 1 (Full), 2 (Half), 4 (Quarter), 6 (Sixth).

***vgopsize*** video group of picture size. Default value is 60. Possible values: 1 (1 Frame Only), 5 (5 Frames), 8 (8 Frames), 10 (10 Frames), 12 (12 Frames), 15 (15 Frames), 25 (25 Frames), 30 (30 Frames), 50 (50 Frames), 60 (60 Frames), 90 (90 Frames), 100 (100 Frames), 120 (120 Frames), 200 (200 Frames), 240 (240 Frames).

***vprotocol*** video protocol.

***vdest*** video destination port.

***vpid*** video packet identifier. Default value is 221.

***vdelay*** video delay. Default value is 300.

***pcrpid*** PCR packet identifier. Default value is 521.

***pcrinterval*** Default value is 50.

***pmtpid*** program map table packet identifier. Default value is 31.

***tsrate*** bit rate for transport stream. Default value is 5000K.

***tslowlat*** transport stream low latency mode. Default value is off. Possible values: on, off.

***feconoff*** Forward error correction in transport stream enable/ disable. Default value is off. Possible values: on, off.

***fecrow*** Forward error correction row. Default value is 1.

***feccol*** Forward error correction column, Default value is 5.

***zixioverhead*** ZIXI Forward Error Correction total overhead as a percentage of transport rate. Default value is 15.

***zixilatency*** ZIXI target latency (to allow ARQ) in milliseconds. Default value is 500.

***zixifecblock*** ZIXI Forward Error Correction block size in milliseconds. Max value is 1/2 of Zixi Latency. Default value is 50.

***zixirateadjen*** ZIXI dynamic bitrate adjustment according to network conditions. Default value is on. Possible values: on, off.

***zixiauthen*** ZIXI dynamic authentication enable/disable . Default value is off. Possible values: on, off.

***zixisession*** ZIXI session id. Default value is test.

***zixiuser*** ZIXI user identifier. Default value is user.

***aenable*** Audio enable/disable. Default value is on. Possible values: on, off.asource. Default value is

MICL.

***apair*** Stereo pair select for digital inputs. Default value is 0. Possible values: (0 (1 2), 1 (3 4), 2 (5 6), 3 (7 8), 260 (1 2,3 4), 548 (1 2,3 4,5 6), 1252 (1 2,3 4,5 6,7 8)).

***acodec*** Audio codec. Default value is fdk_aaclc.

***abitrate*** Audio bit rate. Default value is 128000.

***asamplerate*** Audio sample rate. Default value is 48000.

***aport*** Audio port. Default value is 8700.

***apid*** Audio packet identifier. Default value is 120.

***aptspcr*** Maximum difference between the PTS and PCR in the audio stream Default value is 250.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
StartChannel response: {"ret":"0","status":"RUNNING"}
```

---

## StartMTS

---

Start the MTS.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
StartMTS response: {"ret":"0","status":"OK"}
```

---

## StopChannel

---

Stop the encoder channel and update the state in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

#### Command to execute:

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

#### Example output:

```
StopChannel response: {"ret":"0","status":"STOPPED"}
```

---

## StopMTS

---

Stop the MTS.

##### Parameters:

---

##### Returns:

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

#### Command to execute:

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

#### Example output:

```
StopMTSresponse: {"ret":"0","status":"OK"}
```

---

## TempStatus

---

Get the current temperature status for CPU and lense.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
TempStatus response: {"ret":"0","status":"OK"}
```

---

## update_require_web_login

---

Update/Change whether web login credential required or not in the Database.

**Parameters:**

***require*** Required web login value. Possible values: 1 or 0

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
update_require_web_login response: {"ret":"0","status":"OK"}
```

## UpdatePtz

Update the PTZ value in the Database.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the PTZ values.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=UpdatePtz&rowcount=1&data={"row0":{"db_videosource":"CAMERA","token":
"ptz","name":"videoin1","enabled":1,"pelcoaddr":1,"tcpport":2000,"ptztype":"
relative","flip":"none","pos":"simulated","maxtilt":90,"rowid":1}}
```

---

**Example output:**

```
UpdatePtz response: {"ret":"0","status":"OK"}
```

---

## UpdatePtzPreset

---

Update the PTZ preset value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ preset values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
```

```
passwd admin
```

```
action=UpdatePtzPreset&rowcount=2&data={"row0":{"profiletoken":"profile1","t
oken":"1","name":"home","action":null,"rowid":1},"row1":{"profiletoken":"pro
file1","token":"2","name":"temp","action":null,"rowid":2}}
```

---

**Example output:**

```
UpdatePtzPreset response: {"ret":"0","status":"OK"}
```

---

## UpdatePtzTourSpots

---

Update the PTZ tour spots value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ tour spots values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=UpdatePtzTourSpots&rowcount=1&data={"row0":{"ProfileToken":"profile1"
,"PresetTourToken":"1","PresetDetailToken":"1","StayTime":5,"RowIndex":0,"ac
tion":"blank","rowid":1}}
```

**Example output:**

```
UpdatePtzTourSpots response: {"ret":"0","status":"OK"}
```

## UpdateTerm

Control serial port terminal server for VISCA

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the terminal server values.

***term_mode*** Possible values: client, server

***term_protocol*** Possible values: clearchannel, telnet

***term_localport*** Local TCP port (for server mode)

***term_servaddr*** Remote IP address (for client mode only)

***term_servport*** Remote TCP port (for client mode only)

***term_data_bits*** number of data bits.

***term_parity*** Possible values: None, Odd, Even, Mark, Space

***term_stop_bits*** Stop bits sent at the end of every character.

***term_baudrate*** Possible values: 9600, 19200, 38400, 57600, 115200

***term_devicefile*** device fine name

***term_function*** functionality

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action=UpdateTerm&rowcount=5&data={"row0":{"term_devicefile":"%2Fdev%2FttyHS
1","term_baudrate":"9600","term_data_bits":"8","term_parity":"None","term_st
op_bits":"1","term_function":"rs485","term_localport":"2000","rowid":1},"row
1":{"term_devicefile":"%2Fdev%2FttyHS3","term_baudrate":"9600","term_data_bi
ts":"8","term_parity":"None","term_stop_bits":"1","term_function":"camera1_v
isca","term_localport":"1000","rowid":2},"row2":{"term_devicefile":"%2Fdev%2
FttyHS2","term_baudrate":"9600","term_data_bits":"8","term_parity":"None","t
erm_stop_bits":"1","term_function":"ir_led_zoom_ctrl","term_localport":"1001
","rowid":3},"row3":{"term_devicefile":"%2Fdev%2FttyMSM0","term_baudrate":"1
15200","term_data_bits":"8","term_parity":"None","term_stop_bits":"1","term_
function":"console","term_localport":"1800","rowid":4},"row4":{"term_devicef
ile":"%2Fdev%2FttyHS4","term_baudrate":"2400","term_data_bits":"8","term_par
ity":"None","term_stop_bits":"1","term_function":"user","term_localport":"16
00","rowid":5}}
```

**Example output:**

```
UpdateTerm response: {"ret":"0","status":"OK"}
```

## user_add

Add a new user in the Database.

**Parameters:**

***Name*** Get new user name

***Level*** Get the level of permission for the new user name. Possible value: 0 to 7.

***Password*** Get the password for the new user name.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

**Example output:**

```
user_add response: {"ret":"0","status":"OK"}
```

## user_remove

Remove the user from the database.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the user name.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

```
action: "user_remove", data: '{"Name":"temp"}'
```

---

**Example output:**

```
user_remove response: {"ret":"0","status":"OK"}
```

---

## user_update_level

---

Update/change the level of permission to the given user name.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the user name and level of the permission.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
user_update_level response: {"ret":"0","status":"OK"}
```

---

## user_update_password

---

Update/change the password to the given user name.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the user name and password.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

**Command to execute:**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -
passwd admin
```

---

**Example output:**

```
user_update_password response: {"ret":"0","status":"OK"}
```

---

## download_remove

---

```
http://Encoder_IP/cgi-bin/control.cgi?action=download_remove&data=FILENAME
```

```
http://192.168.0.120/cgi-bin/control.cgi?action=download_remove&data=/media/sda/MOV1_000040.mp4
```

remove the downloaded media content from the usb drive or sd card.

**Parameters:**

---

***data*** Filename from the downloaded media content to be removed from the removable media device.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

# Generic Control (GET)

## 8021x

---

Get the exthernet 802x information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***interface*** TEXT,

***enable*** TEXT.

***eap*** TEXT.

***anonymous_identity*** TEXT.

***identity*** TEXT.

***password*** TEXT.

***private_key_password*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
8021x response:
{"anonymous_identity":"","ca_cert_status":"Unavailable","eap":"peap","enable
":"off","identity":"","interface":"eth0","password":"","priv_cert_status":"U
navailable","priv_key_status":"Unavailable","private_key_password":"","ret":
"0"}
```

---

## GetCronJobs

---

Get the schedule job from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***enable*** INTEGER,

***cron*** TEXT.

***func*** TEXT.

***func_args*** TEXT.

***type*** TEXT.

***minutes*** INTEGER.

***hours*** INTEGER.

***day*** INTEGER.

***month*** INTEGER.

***cron*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
GetCronJobs response:
{"data":[],"metadata":[{"datatype":"integer","editable":true,"label":"Enable
","name":"enable"},{"datatype":"text","editable":true,"label":"Name","name":
"name"},{"datatype":"text","editable":true,"label":"Function","name":"func"}
,{"datatype":"text","editable":true,"label":"Function
Arguements","name":"func_args"},{"datatype":"text","editable":true,"label":"
Type","name":"type"},{"datatype":"integer","editable":true,"label":"Minutes"
,"name":"minutes"},{"datatype":"integer","editable":true,"label":"Hours","na
me":"hours"},{"datatype":"integer","editable":true,"label":"Day","name":"day
"},{"datatype":"integer","editable":true,"label":"Month","name":"month"},{"d
atatype":"text","editable":true,"label":"Cron
Command","name":"cron"}],"ret":"0"}
```

---

## GetNFSMount

---

Get the Network File Sharing mounted infromation from the device.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mounted*** TEXT,

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Get NFS Mount response: {"mounted":256,"ret":"0"}
```

---

## aspect_info

---

Get the video resolution for the current channel from the Database and calculate the aspect ratio.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***numerator*** INTEGER,

***denominator*** INTEGER.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Aspect Info response:
{"denominator":9,"numerator":16,"ret":"0","status":"OK"}
```

## boardinfo

Get the board and model information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***logo_enable*** TEXT,

***logo_filename*** TEXT.

***logo_width*** INTEGER.

***logo_height*** INTEGER.

***logo_blob*** BLOB.

***ico_blob*** BLOB.

***model_enable*** TEXT.

***model_name*** TEXT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Boardinfo response:
{"MODEL":"FV4K-13A","board_id":"0xFF21","hw50serial":"50014401222610007","hw
serial":"30014410221610012","hwversion":"FV4K-13A","logo_enable":"off","logo
_height":156,"logo_width":319,"macaddr":"40:cd:3a:06:10:bc","model_enable":"
off","model_name":"Z3-
Encoder","opmode":"encoder","opstate":"RUNNING","processor_id":"cv22bub","re
t":"0","sensor_serial":"30014410221610012\n","sysdevicename":"Z3Cam","z3_avm
ux":"enabled","z3_klv":"enabled","z3_klv_serial_only":"enabled","z3_sntp":"e
nabled","z3_termsrv":"enabled","z3_tslowlat":"enabled","z3_webproxy":"enable
d"}
```

## cam_state

Get the current camera state information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT,

***enc_channels*** TEXT.

***opmode*** TEXT.

***enc_current_preset*** TEXT.

***dec_current_preset*** TEXT.

---

**Command to execute and Example :**

Please refer in this document in the below section cam_state (sys_DIGEST_ARGPARSE.py)

---

## camera

---

Get the current camera settings from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***zoom_direct_value*** INTEGER.

***white_balance_mode*** TEXT.

***color_gain*** INTEGER.

***color_hue*** INTEGER.

***chroma_suppress*** INTEGER.

***wb_manual_rgain_direct*** INTEGER.

***wb_manual_bgain_direct*** INTEGER.

***visca_localport*** TEXT.

***optical_zoom_only*** BOOL. 0 or 1.

***focus_direct_value*** INTEGER.

***manual_focus*** TEXT.

***flexio_localport*** TEXT.

***exposure_mode*** INTEGER.

***shutter*** INTEGER.

***iris*** INTEGER.

***gain*** INTEGER.

***high_sensitivity*** INTEGER.

***hlc_level*** INTEGER.

***hlc_level_mask*** INTEGER.

***stable_zoom*** INTEGER.

***eflip*** INTEGER.

***lr_reverse*** INTEGER.

***monitor_mode*** TEXT.

***genlock_source*** INTEGER.

***manual_icr*** TEXT.

***zoom_step_size*** INTEGER.

***focus_step_size*** INTEGER.

***img_freeze*** INTEGER.

***hr_mode*** INTEGER.

***img_stabilizer*** INTEGER.

***img_bw*** INTEGER.

***nr_2d_level*** INTEGER.

***nr_3d_level*** INTEGER.

***icr_threshold*** INTEGER.

***slow_shutter*** BOOL. 0 or 1.

***slow_shutter_limit*** INTEGER.

***flicker_reduction*** BOOL. 0 or 1.

***img_stabilizer_level*** INTEGER.

***wide_dynamic_range*** TEXT.

***ve_brightness*** TEXT.

***ve_compensation_type*** TEXT.

***ve_compensation_level*** TEXT.

***tab_index*** TEXT.

***imgvflip*** INTEGER.

***imghflip*** INTEGER.

***imgrotate*** INTEGER.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Camera response:
{"backlight":3,"chroma_suppress":1,"color_gain":4,"color_hue":7,"dbversion":
"1.1","eflip":3,"exposure_mode":0,"flexio_localport":"1500","flicker_reducti
on":0,"focus_direct_value":4096,"focus_step_size":1,"gain":4,"genlock_source
":2,"high_sensitivity":3,"hlc_level":0,"hlc_level_mask":0,"hr_mode":3,"icr_t
hreshold":-1,"img_bw":0,"img_freeze":3,"img_stabilizer":3,"img_stabilizer_le
vel":0,"imghflip":0,"imgrotate":0,"imgvflip":0,"iris":15,"lr_reverse":3,"man
ual_focus":"auto","manual_icr":"off","monitor_mode":"2160p-29.97","nr_2d_lev
el":5,"nr_3d_level":4,"optical_zoom_only":0,"ret":"0","shutter":7,"slow_shut
ter":0,"slow_shutter_limit":-1,"stable_zoom":0,"tab_index":"1","ve_brightnes
s":"-1","ve_compensation_level":"-1","ve_compensation_type":"-1","visca_loca
lport":"1000","wb_manual_bgain_direct":190,"wb_manual_rgain_direct":190,"whi
te_balance_mode":"auto","wide_dynamic_range":"3","zoom_direct_value":0,"zoom
_step_size":4}
```

---

## camera_exposure

---

Get the current camera exposure values from the visca camera.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***ae_mode*** TEXT,

***shutter*** TEXT.

***iris*** TEXT.

***gain_inquiry*** TEXT.

***sensitivity*** TEXT.

***hls_inquiry*** TEXT.

***min_shutter*** INTEGER.

***max_shutter*** INTEGER.

***shutter_label*** TEXT.

***min_iris*** INTEGER.

***max_iris*** INTEGER.

***iris_label*** TEXT.

***min_gain*** INTEGER.

***max_gain*** INTEGER.

***gain_label*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Camera Exposure response:
{"ae_mode":0,"backlight":3,"gain":10,"gain_label":["0db",null,null,null,null
,null,null,null,"24db",null,null,null,null,null,null,null,"48db"],"hlc":0,"i
ris":25,"iris_label":["0",null,null,null,null,null,null,null,null,null,null,
null,null,null,null,"F4.8",null,null,null,null,null,null,null,null,"F2.0"],"
max_gain":17,"max_iris":25,"max_shutter":33,"min_gain":1,"min_iris":0,"min_s
hutter":6,"ret":"0","sensitivity":3,"shutter":16,"shutter_label":["1/1",null
,null,null,null,null,null,null,null,null,null,null,null,null,"1/100",null,nu
ll,null,null,null,null,null,null,null,null,null,"1/10K"]}
```

## camera_monitor_mode

Get the current camera monitor mode values from the visca camera.

**Parameters:**

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***modes*** TEXT.

***current_mode*** TEXT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Camera Monitor Mode response:
{"current_mode":"2160p-29.97","modes":{"1080p-25":8,"1080p-29.97":6,"1080p-50":20,"1080p-59.94":19,"2160p-25":30,"2160p-29.97":29,"720p-50":12,"720p-59.94":9},"ret":"0"}
```

---

## dec

---

Get the current decoder settings from the Database.

**Parameters:**

---

***chn*** Get the decoder channel number. The default decoder channel is 1. Select between decoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT.

***channel*** INTEGER.

***url*** TEXT.

***aenable*** TEXT.

***viewport*** TEXT.

***latency_mode*** TEXT.

***max_latency_ms*** TEXT.

***srt_mode*** TEXT.

***srt_latency*** TEXT.

***srt_decrypt*** TEXT.

***srt_pass*** TEXT.

***zixi_fec*** TEXT.

***zixi_fecoverhead*** TEXT.

***zixi_fecblock*** TEXT.

***zixi_latency*** TEXT.

***zixi_decrypt*** TEXT.

***zixi_pass*** TEXT.

***rtsp_flags*** TEXT.

---

## download

---

Get the downloaded file information from the removable storage media.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***job*** Optional***.*** The job from the schedule tab.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***FileName*** TEXT.

***FileSize*** TEXT.

***LastModified*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Download response:
{"data":[],"metadata":[{"datatype":"text","editable":false,"label":"File
Name","name":"download_filename"},{"datatype":"integer","editable":false,"la
bel":"File
Size","name":"download_size"},{"datatype":"text","editable":false,"label":"L
ast Modified","name":"download_lastmodified"}],"ret":"0"}
```

---

## drs_tamarisk_settings

---

Get the camera model of drs tamarisk settings fromt the Database.

**Parameters:**

---

***job*** Optional***.*** The job from the schedule tab.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***color_pallette*** TEXT.

***tab_index*** TEXT.

***color_pallette_disabled*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
DRS Tamarisk Settings response:
{"color_pallette":"white_hot","color_pallette_disabled":"false","ret":"0","t
ab_index":"1"}
```

---

## enc

---

Get the current channel encoder settings from the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***preset*** TEXT,

***channel*** INTEGER.

***vbitrate*** TEXT.

***vframeratediv*** INTEGER.

***vsource*** TEXT.

***vcodec*** TEXT.

***vprofile*** TEXT.

***vprotocol*** TEXT.

***avmux_index*** TEXT.

***vgdr*** TEXT.

***vdest*** TEXT.

***aenable*** TEXT.

***acodec*** TEXT.

***abitrate*** TEXT.

***asamplerate*** TEXT.

***aport*** TEXT.

***apair*** INTEGER.

***vres*** TEXT.

***vgopsize*** INTEGER.

***vpid*** INTEGER.

***apid*** INTEGER.

***zixiauthen*** TEXT.

***zixisession*** TEXT.

***zixiuser*** TEXT.

***zixioverhead*** TEXT.

***zixifecblock*** TEXT.

***zixilatency*** TEXT.

***zixirateadjen*** TEXT.

***fecrow*** INTEGER.

***feccol*** INTEGER.

***feconoff*** TEXT.

***aptspcr*** INTEGER.

***tslowlat*** TEXT.

***tsrate*** TEXT.

***pcrpid*** INTEGER.

***pcrinterval*** INTEGER.

***pmtpid*** INTEGER.

***klvenable*** TEXT.

***klvmode*** TEXT.

***klvmuxmethod*** TEXT.

***klvsrc*** TEXT.

***klvbrate*** TEXT.

***klvserialbaud*** TEXT.

***klvpid*** TEXT.

***vratectrl*** TEXT.

***vdelay*** INTEGER.

***storage*** TEXT.

***fprefix*** TEXT.

***asource*** TEXT.

***authonoff*** TEXT.

***auth_user*** TEXT.

***auth_passwd*** TEXT.

***auxonoff*** TEXT.

***filesize*** TEXT.

***nfstrength*** TEXT.

***telopenable*** TEXT.

***teloptext*** TEXT.

***teloplocation*** TEXT.

***telopcharsize*** TEXT.

***teloptextcolor*** TEXT.

***telopoutlineenable*** TEXT.

***telopoutlinecolor*** TEXT.

***gps_overlay_enable*** TEXT.

***gps_overlay_device*** TEXT.

***gps_overlay_location*** TEXT.

***gps_overlay_char_size*** TEXT.

***pipenable*** TEXT.

***piplocation*** TEXT.

***vquality*** TEXT.

***vinterlacemode*** TEXT.

***vmulticastdest*** TEXT.

***amulticastdest*** TEXT.

***rtsp_auth_enable*** TEXT.

***rtsp_auth_username*** TEXT.

***rtsp_auth_password*** TEXT.

***rtsp_transport_mode*** TEXT.

***lowdelay_opt*** TEXT.

***vcropaspect*** TEXT.

***vcrop_enable*** TEXT.

***vcrop_width*** INTEGER.

***vcrop_height*** INTEGER.

***vcrop_x*** INTEGER.

***vcrop_y*** INTEGER.

***rotate_enable*** INTEGER.

***rotate_angle*** INTEGER.

***rtmp265_enable*** TEXT.

***srt_pass*** TEXT.

***srt_encrypt*** INTEGER.

***srt_mode*** INTEGER.

***srt_destAddr*** TEXT.

***mmulticastdest*** TEXT.

***mport*** TEXT.

***frame_loss_mode*** TEXT.

***frame_loss_gap*** INTEGER.

***frame_loss_overshot*** INTEGER.

***mounts*** TEXT.

***source_status_str*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Encoder response:
{"abitrate":"128000","acodec":"fdk_aaclc","aenable":"on","amulticastdest":"2
25.1.2.3","apair":0,"apid":120,"aport":"8700","aptspcr":250,"asamplerate":"1
","asource":"MIC","auth_passwd":"password","auth_user":"user","authonoff":"o
ff","auxonoff":"off","avmux_index":"streaming","channel":1,"enc_status":"RUN
NING","feccol":5,"feconoff":"off","fecrow":1,"filesize":"1024M","fprefix":"M
OV1_%C","frame_loss_gap":0,"frame_loss_mode":"none","frame_loss_overshot":10
0,"gps_overlay_char_size":"32","gps_overlay_device":"/dev/ttyS1","gps_overla
y_enable":"off","gps_overlay_location":"top_right","klvbrate":"1000","klvena
ble":"off","klvmode":"sdi","klvmuxmethod":"sync","klvpid":"35","klvserialbau
d":"115200","klvsrc":"/dev/ttyS1","lowdelay_opt":"off","mmulticastdest":"225
.1.2.3","mounts":"/dev/mmcblk0p1
/media/mmcblk0p1/,","mport":"8800","nfstrength":"0","pcrinterval":50,"pcrpid
":521,"pipenable":"off","piplocation":"top_right","pmtpid":31,"preset":"actv
_preset","ret":"0","rotate_angle":0,"rotate_enable":"off","rtmp265_enable":"
off","rtsp_auth_enable":"off","rtsp_auth_password":"admin","rtsp_auth_userna
me":"admin","rtsp_transport_mode":"all","source_status_str":"+CAMERA
3840x2160p 29.97
fps\n","srt_destAddr":"192.168.0.6","srt_encrypt":0,"srt_mode":0,"srt_pass":
"password1234","storage":"/media/sda1","telopcharsize":"32","telopenable":"o
ff","teloplocation":"top_left","telopoutlinecolor":"0xFF000000","telopoutlin
eenable":"off","teloptext":"Z3Cam","teloptextcolor":"0xFFFFFFFF","tslowlat":
"on","tsrate":"3000K","vbitrate":"6M","vcodec":"h265","vcrop_enable":"off","
vcrop_height":1080,"vcrop_width":1920,"vcrop_x":0,"vcrop_y":0,"vcropaspect":
"off","vdelay":1000,"vdest":"192.168.0.6:8600","vframeratediv":1,"vgdr":"off
","vgopsize":60,"vinterlacemode":"combine","vmulticastdest":"225.1.2.3","vpi
d":221,"vprofile":"high","vprotocol":"rtsp","vquality":"balanced","vratectrl
":"vbr","vres":"follow_input","vsource":"CAMERA","zixiauthen":"off","zixifec
block":"50","zixilatency":"500","zixioverhead":"15","zixirateadjen":"on","zi
```

xisession":"test","zixiuser":"user"}

---

## filepicker

---

Get the file type and its information from the removable storage media.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***filepicker_filename*** TEXT,

***filepicker_size*** INTEGER.

***filepicker_lastmodified*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Filepicket response:
{"data":[],"metadata":[{"datatype":"text","editable":false,"label":"File
Name","name":"filepicker_filename"},{"datatype":"integer","editable":false,"
label":"File
Size","name":"filepicker_size"},{"datatype":"text","editable":false,"label":
"Last Modified","name":"filepicker_lastmodified"}],"ret":"0"}
```

## fmt

Get the removable storage infromation from the device using fdisk command.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mounts*** TEXT,

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Format response: {"mounts":"mmcblk0p1\n
/dev/mmcblk0,","ret":"0","store_logs":"off"}
```

## focusStepSize

Get the current camera focus step values from the Database.

**Parameters:**

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***focus_step_size*** INTEGER,

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Focus Step Size response: {"ret":"0","size":1}
```

## fpga

Get the current fpga from firmware filename.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***fpgafileglob*** TEXT,

***options*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
FPGA response:
{"fpgafileglob":"fpga_none.bin|fpga2_0139_cvbs.bin","fpgaoptions":{"Boson
only        ":"fpga_unused.bin|fpga2_0139_boson.bin","Composite only
":"fpga_unused.bin|fpga2_0139_cvbs.bin","HDMI 4K| Boson
":"fpga_none.bin|fpga2_0139_boson.bin","HDMI 4K|
Composite":"fpga_none.bin|fpga2_0139_cvbs.bin","HDMI 4K| SonyLVDS
":"fpga_none.bin|fpga2_0139.bin","HDMI 4K|
Tamarisk":"fpga_none.bin|fpga2_0139_tamarisk.bin","HDMI 4K| Tau2
":"fpga_none.bin|fpga2_0139_tau2.bin","HDMI 4K| Tenum
":"fpga_none.bin|fpga2_0139_tenum.bin","HDMI 4K| Unused
":"fpga_none.bin|fpga2_none.bin","Micro-HDMI| Boson
":"fpga_none_microhdmi.bin|fpga2_0139_boson.bin","Micro-HDMI|
Composite":"fpga_none_microhdmi.bin|fpga2_0139_cvbs.bin","Micro-HDMI|
SonyLVDS ":"fpga_none_microhdmi.bin|fpga2_0139.bin","Micro-HDMI| Tamarisk
":"fpga_none_microhdmi.bin|fpga2_0139_tamarisk.bin","Micro-HDMI| Tau2
":"fpga_none_microhdmi.bin|fpga2_0139_tau2.bin","Micro-HDMI| Tenum
":"fpga_none_microhdmi.bin|fpga2_0139_tenum.bin","Micro-HDMI| Unused
":"fpga_none_microhdmi.bin|fpga2_none.bin","SonyLVDS only
":"fpga_unused.bin|fpga2_0139.bin","Tamarisk only
":"fpga_unused.bin|fpga2_0139_tamarisk.bin","Tau2  only
":"fpga_unused.bin|fpga2_0139_tau2.bin","Tenum only
":"fpga_unused.bin|fpga2_0139_tenum.bin"},"ret":"0"}
```

---

## history

---

Get the decoder history infromation from the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder

channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***channel*** TEXT,

***histidx*** TEXT.

***url*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
History response: {"ret":"0","status":"OK"}
```

---

## ipinfo

---

Get the internet protocol infromation from the board.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***local_ip*** TEXT,

***local_netmask*** TEXT.

***default_gw*** TEXT.

***ipmtu*** TEXT.

***eth_speed*** TEXT.

***eth_duplex*** TEXT.

***do_autostart*** BOOLEAN.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
IP Info response:
{"default_gw":"192.168.10.1","do_autostart":1,"eth_duplex":"AUTO","eth_speed
":"AUTO","ipmtu":1500,"local_dnsip":"8.8.8.8","local_dnsip2":"8.8.4.4","loca
l_hostname":"z3-
he4k","local_ip":"192.168.10.127","local_netmask":"255.255.255.0","ret":"0",
"use_dhcp":"1"}
```

---

## onvif

---

Get the ONVIF miscallenous settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***list_all_video_sources*** INTEGER,

***dns_from_dhcp*** INTEGER,

***ntp_from_dhcp*** INTEGER,

***ntp_enable*** INTEGER,

***daylightsaving*** INTEGER,

***timezone*** TEXT.

***http_enable*** INTEGER,

***http_port*** INTEGER,

***https_enable*** INTEGER,

***https_port*** INTEGER,

***rtsp_enable*** INTEGER,

***rtsp_port*** INTEGER,

***discoverable*** INTEGER,

***ntp*** TEXT.

***dns_search*** TEXT.

***ptz_timeout*** TEXT.

***primary fixed_profile_max*** INTEGER,

***secondary fixed_profile_max*** INTEGER,

***en_persistent_audio*** TEXT.

***onvif_enable*** BOOLEAN.

***vmd_enable*** BOOLEAN.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
ONVIF response:
{"daylightsaving":"0","discoverable":"1","dns_from_dhcp":"1","dns_search":"l
ocaldomain","en_persistent_audio":"1","fixed_profile_max":"1","http_enable":
"1","http_port":"80","https_enable":"0","https_port":"443","list_all_video_s
ources":"0","ntp":"pool.ntp.org","ntp_enable":"1","ntp_from_dhcp":"0","onvif
_enable":"on","ptz_timeout":10,"ret":"0","rtsp_enable":"1","rtsp_port":"554"
,"timezone":"TaipeiStandardTime-8","vmd_enable":"off"}
```

---

## onvif_vmd

---

Get the ONVIF video motion detection and too darkness settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***vmd_sens*** INTEGER,

***vmd_zone_modify*** TEXT.

***vmd_vcrop_width*** INTEGER.

***vmd_vcrop_height*** INTEGER.

***vmd_vcrop_x*** INTEGER.

***vmd_vcrop_y*** INTEGER.

***td_enable*** TEXT.

***td_thresh*** INTEGER.

***vmd_td_channel*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
ONVIF VMD response:
{"ret":"0","td_enable":"off","td_thresh":85,"vmd_enable":"off","vmd_sens":1,
"vmd_td_channel":"CH1","vmd_vcrop_height":0,"vmd_vcrop_width":0,"vmd_vcrop_x
":0,"vmd_vcrop_y":0,"vmd_zone_modify":"off"}
```

---

## permission

---

Get the current permission settings for admin, operators, and users from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***username*** TEXT,

***userlevel*** TEXT.

***api_command*** TEXT.

***superadmin*** TEXT.

**admin** TEXT.

***operator1*** TEXT.

***operator2*** TEXT.

***user1*** TEXT.

***user2*** TEXT.

***user3*** TEXT.

***anonymous*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
PERMISSIONS response:
{"permissions":{"Z38021xImportHandler:post":"W","Z3DebugHandler:get":"R","Z3
DownloadMediaHandler:get":"R","Z3ExportHandler:get":"R","Z3FWUploadHandler:p
ost":"W","Z3ImportHandler:post":"W","Z3LogoHandler:get":"?","Z3LogsHandler:g
et":"R","Z3RequestHandler:get:8021x":"?","Z3RequestHandler:get:GetCronJobs":
"R","Z3RequestHandler:get:GetNFSMount":"?","Z3RequestHandler:get:aspect_info
":"?","Z3RequestHandler:get:boardinfo":"R","Z3RequestHandler:get:cam_state":
"R","Z3RequestHandler:get:camera":"?","Z3RequestHandler:get:camera_exposure"
:"?","Z3RequestHandler:get:camera_monitor_mode":"?","Z3RequestHandler:get:de
c":"R","Z3RequestHandler:get:download":"?","Z3RequestHandler:get:drs_tamaris
k_settings":"?","Z3RequestHandler:get:enc":"R","Z3RequestHandler:get:filepic
ker":"?","Z3RequestHandler:get:fmt":"W","Z3RequestHandler:get:focusStepSize"
:"?","Z3RequestHandler:get:fpga":"?","Z3RequestHandler:get:history":"R","Z3R
equestHandler:get:ipinfo":"R","Z3RequestHandler:get:onvif":"?","Z3RequestHan
dler:get:onvif_vmd":"?","Z3RequestHandler:get:permission":"R","Z3RequestHand
ler:get:ptz":"?","Z3RequestHandler:get:ptz_preset":"?","Z3RequestHandler:get
:ptz_tour":"?","Z3RequestHandler:get:snmp":"?","Z3RequestHandler:get:ssl":"?
","Z3RequestHandler:get:stats":"?","Z3RequestHandler:get:sys":"R","Z3Request
Handler:get:term":"?","Z3RequestHandler:get:users":"R","Z3RequestHandler:get
:video_group":"?","Z3RequestHandler:get:wifi":"?","Z3RequestHandler:get:zoom
StepSize":"?","Z3RequestHandler:post:AStreamStatus":"?","Z3RequestHandler:po
st:AddChannel":"W","Z3RequestHandler:post:CameraControl:SetColorPallette":"W
","Z3RequestHandler:post:CameraControl:ae_mode":"W","Z3RequestHandler:post:C
ameraControl:ae_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:agc_f
ilter":"W","Z3RequestHandler:post:CameraControl:agc_midpoint":"W","Z3Request
Handler:post:CameraControl:agc_type":"W","Z3RequestHandler:post:CameraContro
l:auto_icr_disable":"W","Z3RequestHandler:post:CameraControl:auto_icr_enable
":"W","Z3RequestHandler:post:CameraControl:autocal_disable":"W","Z3RequestHa
ndler:post:CameraControl:autocal_enable":"W","Z3RequestHandler:post:CameraCo
ntrol:backlight_inquiry":"R","Z3RequestHandler:post:CameraControl:backlight_
off":"W","Z3RequestHandler:post:CameraControl:backlight_on":"W","Z3RequestHa
ndler:post:CameraControl:black_hot":"W","Z3RequestHandler:post:CameraControl
:brightness":"W","Z3RequestHandler:post:CameraControl:brightness_bias":"W","
```

Z3RequestHandler:post:CameraControl:brightness_inquiry":"R","Z3RequestHandler:post:CameraControl:cam_control_inquiry":"R","Z3RequestHandler:post:CameraControl:cam_custom_recall":"W","Z3RequestHandler:post:CameraControl:cam_custom_reset":"W","Z3RequestHandler:post:CameraControl:cam_custom_set":"W","Z3RequestHandler:post:CameraControl:chroma_get_suppress":"R","Z3RequestHandler:post:CameraControl:chroma_suppress":"W","Z3RequestHandler:post:CameraControl:color_disable":"W","Z3RequestHandler:post:CameraControl:color_enable":"W","Z3RequestHandler:post:CameraControl:color_gain":"W","Z3RequestHandler:post:CameraControl:color_get_gain":"R","Z3RequestHandler:post:CameraControl:color_get_hue":"R","Z3RequestHandler:post:CameraControl:color_hue":"W","Z3RequestHandler:post:CameraControl:color_select":"W","Z3RequestHandler:post:CameraControl:colorlut_arctic":"W","Z3RequestHandler:post:CameraControl:colorlut_blackhot":"W","Z3RequestHandler:post:CameraControl:colorlut_disable":"W","Z3RequestHandler:post:CameraControl:colorlut_enable":"W","Z3RequestHandler:post:CameraControl:colorlut_globow":"W","Z3RequestHandler:post:CameraControl:colorlut_gradedfire":"W","Z3RequestHandler:post:CameraControl:colorlut_hottest":"W","Z3RequestHandler:post:CameraControl:colorlut_ironbow":"W","Z3RequestHandler:post:CameraControl:colorlut_lava":"W","Z3RequestHandler:post:CameraControl:colorlut_rainbow":"W","Z3RequestHandler:post:CameraControl:colorlut_rainbow_hc":"W","Z3RequestHandler:post:CameraControl:colorlut_whitehot":"W","Z3RequestHandler:post:CameraControl:contrast_adjust_level":"W","Z3RequestHandler:post:CameraControl:contrast_adjust_level_inquiry":"R","Z3RequestHandler:post:CameraControl:contrast_inquiry":"R","Z3RequestHandler:post:CameraControl:defog_inquiry":"R","Z3RequestHandler:post:CameraControl:defog_on_direct":"W","Z3RequestHandler:post:CameraControl:display_off":"W","Z3RequestHandler:post:CameraControl:display_on":"W","Z3RequestHandler:post:CameraControl:display_on_off":"W","Z3RequestHandler:post:CameraControl:do_ffc":"W","Z3RequestHandler:post:CameraControl:dvo_set_output_format_ycbcr":"W","Z3RequestHandler:post:CameraControl:dzoom_combine_mode":"W","Z3RequestHandler:post:CameraControl:dzoom_direct":"W","Z3RequestHandler:post:CameraControl:dzoom_inquiry":"R","Z3RequestHandler:post:CameraControl:dzoom_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:dzoom_off":"W","Z3RequestHandler:post:CameraControl:dzoom_on":"W","Z3RequestHandler:post:CameraControl:dzoom_separage_mode":"W","Z3RequestHandler:post:CameraControl:dzoom_separate_mode":"W","Z3RequestHandler:post:CameraControl:dzoom_stop":"W","Z3RequestHandler:post:CameraControl:dzoom_super_res":"W","Z3RequestHandler:post:CameraControl:dzoom_tele_var":"W","Z3RequestHandler:post:CameraControl:dzoom_wide_var":"W","Z3RequestHandler:post:CameraControl:dzoom_x1_max":"W","Z3RequestHandler:post:CameraControl:e_pan_tilt_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:e_pan_tilt_off":"W","Z3RequestHandler:post:CameraControl:e_pan_tilt_on":"W","Z3RequestHandler:post:CameraControl:eflip_inquiry":"R","Z3RequestHandler:post:CameraControl:eflip_off":"W","Z3RequestHandler:post:CameraControl:eflip_on":"W","Z3RequestHandler:post:CameraControl:expcomp_direct":"W","Z3RequestHandler:post:CameraControl:expcomp_down":"W","Z3RequestHandler:post:CameraControl:expcomp_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:expcomp_off":"W","Z3RequestHandler:post:CameraControl:expcomp_on":"W","Z3RequestHandler:post:CameraControl:expcomp_pos_inquiry":"R","Z3RequestHandler:post:CameraControl:expcomp_reset":"W","Z3RequestHandler:post:CameraControl:expcomp_up":"W","Z3RequestHandler:post:CameraControl:ffc_period":"W","Z3RequestHandler:post:CameraControl:ffc_temp_delta":"W","Z3RequestHandler:post:CameraControl:ffc_warn_time":"W","Z3RequestHandler:post:CameraControl:flicker_detection_inquiry":"R","Z3Requ

estHandler:post:CameraControl:flicker_reduction_inquiry":"R","Z3RequestHandler:post:CameraControl:flicker_reduction_off":"W","Z3RequestHandler:post:CameraControl:flicker_reduction_on":"W","Z3RequestHandler:post:CameraControl:focus_auto":"W","Z3RequestHandler:post:CameraControl:focus_direct":"W","Z3RequestHandler:post:CameraControl:focus_get_mode":"R","Z3RequestHandler:post:CameraControl:focus_get_pos":"R","Z3RequestHandler:post:CameraControl:focus_get_pos_meter":"R","Z3RequestHandler:post:CameraControl:focus_manual":"W","Z3RequestHandler:post:CameraControl:focus_near_limit":"W","Z3RequestHandler:post:CameraControl:focus_near_limit_inquiry":"R","Z3RequestHandler:post:CameraControl:focus_set_pos_meter":"W","Z3RequestHandler:post:CameraControl:focus_stop":"W","Z3RequestHandler:post:CameraControl:focus_tele_std":"W","Z3RequestHandler:post:CameraControl:focus_tele_var":"W","Z3RequestHandler:post:CameraControl:focus_toggle":"W","Z3RequestHandler:post:CameraControl:focus_wide_std":"W","Z3RequestHandler:post:CameraControl:focus_wide_var":"W","Z3RequestHandler:post:CameraControl:fpa_temp_inquiry":"R","Z3RequestHandler:post:CameraControl:gain":"W","Z3RequestHandler:post:CameraControl:gain_inquiry":"R","Z3RequestHandler:post:CameraControl:gain_mode":"W","Z3RequestHandler:post:CameraControl:get_camera_framerate":"R","Z3RequestHandler:post:CameraControl:get_camera_part_number":"R","Z3RequestHandler:post:CameraControl:get_camera_serial_number":"R","Z3RequestHandler:post:CameraControl:get_color_tables":"R","Z3RequestHandler:post:CameraControl:get_colorlut_current":"R","Z3RequestHandler:post:CameraControl:get_colorlut_state":"R","Z3RequestHandler:post:CameraControl:get_fpa_temp_celsius":"R","Z3RequestHandler:post:CameraControl:get_low_delay_mode":"R","Z3RequestHandler:post:CameraControl:get_revision":"R","Z3RequestHandler:post:CameraControl:get_sensor_part_number":"R","Z3RequestHandler:post:CameraControl:get_sensor_serial_number":"R","Z3RequestHandler:post:CameraControl:get_software_rev":"R","Z3RequestHandler:post:CameraControl:get_tnr_state":"R","Z3RequestHandler:post:CameraControl:get_zoom_limits_mm":"R","Z3RequestHandler:post:CameraControl:high_sensitivity_inquiry":"R","Z3RequestHandler:post:CameraControl:high_sensitivity_off":"W","Z3RequestHandler:post:CameraControl:high_sensitivity_on":"W","Z3RequestHandler:post:CameraControl:hlc":"W","Z3RequestHandler:post:CameraControl:hlc_inquiry":"R","Z3RequestHandler:post:CameraControl:hr_inquiry":"R","Z3RequestHandler:post:CameraControl:hr_off":"W","Z3RequestHandler:post:CameraControl:hr_on":"W","Z3RequestHandler:post:CameraControl:ice_disable":"W","Z3RequestHandler:post:CameraControl:ice_enable":"W","Z3RequestHandler:post:CameraControl:icr_auto_inquiry":"R","Z3RequestHandler:post:CameraControl:icr_mode":"W","Z3RequestHandler:post:CameraControl:icr_mode_inquiry":"R","Z3RequestHandler:post:CameraControl:icr_threshold_inquiry":"R","Z3RequestHandler:post:CameraControl:image_freeze_inquiry":"R","Z3RequestHandler:post:CameraControl:image_freeze_off":"W","Z3RequestHandler:post:CameraControl:image_freeze_on":"W","Z3RequestHandler:post:CameraControl:img_blackwhite_inquiry":"R","Z3RequestHandler:post:CameraControl:img_blackwhite_off":"W","Z3RequestHandler:post:CameraControl:img_blackwhite_on":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_hold":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_inquiry":"R","Z3RequestHandler:post:CameraControl:img_stabilizer_level":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_level_inquiry":"R","Z3RequestHandler:post:CameraControl:img_stabilizer_off":"W","Z3RequestHandler:post:CameraControl:img_stabilizer_on":"W","Z3RequestHandler:post:CameraControl:iris":"W","Z3RequestHandler:post:CameraControl:iris_inquiry":"R","Z3RequestHandler:post:CameraControl:lens_control_inquiry":"R","Z3RequestHandler:post:CameraControl:lens_get_temp":"

R","Z3RequestHandler:post:CameraControl:lr_reverse_inquiry":"R","Z3RequestHandler:post:CameraControl:lr_reverse_off":"W","Z3RequestHandler:post:CameraControl:lr_reverse_on":"W","Z3RequestHandler:post:CameraControl:min_shutter_inquiry":"R","Z3RequestHandler:post:CameraControl:min_shutter_limit":"W","Z3RequestHandler:post:CameraControl:min_shutter_limit_inquiry":"R","Z3RequestHandler:post:CameraControl:min_shutter_off":"W","Z3RequestHandler:post:CameraControl:min_shutter_on":"W","Z3RequestHandler:post:CameraControl:mute_on_off":"W","Z3RequestHandler:post:CameraControl:noise_2d3d":"W","Z3RequestHandler:post:CameraControl:noise_2d3d_inquiry":"R","Z3RequestHandler:post:CameraControl:noise_reduction":"W","Z3RequestHandler:post:CameraControl:noise_reduction_inquiry":"R","Z3RequestHandler:post:CameraControl:part_number_inquiry":"R","Z3RequestHandler:post:CameraControl:power_off":"W","Z3RequestHandler:post:CameraControl:power_on":"W","Z3RequestHandler:post:CameraControl:raw":"W","Z3RequestHandler:post:CameraControl:reboot_camera":"W","Z3RequestHandler:post:CameraControl:register_read":"R","Z3RequestHandler:post:CameraControl:register_write":"W","Z3RequestHandler:post:CameraControl:run_ffc":"W","Z3RequestHandler:post:CameraControl:serial_number_inquiry":"R","Z3RequestHandler:post:CameraControl:set_low_delay_mode":"W","Z3RequestHandler:post:CameraControl:set_monitor_mode":"W","Z3RequestHandler:post:CameraControl:shutter":"W","Z3RequestHandler:post:CameraControl:shutter_inquiry":"R","Z3RequestHandler:post:CameraControl:shutter_position":"R","Z3RequestHandler:post:CameraControl:shutter_temp":"R","Z3RequestHandler:post:CameraControl:shutter_temp_inquiry":"R","Z3RequestHandler:post:CameraControl:slow_shutter":"W","Z3RequestHandler:post:CameraControl:slow_shutter_inquiry":"R","Z3RequestHandler:post:CameraControl:slow_shutter_limit_inquiry":"R","Z3RequestHandler:post:CameraControl:spot_display":"W","Z3RequestHandler:post:CameraControl:spot_display_inquiry":"W","Z3RequestHandler:post:CameraControl:spot_meter_data_advanced":"W","Z3RequestHandler:post:CameraControl:spot_meter_data_inquiry":"R","Z3RequestHandler:post:CameraControl:spot_meter_inquiry":"R","Z3RequestHandler:post:CameraControl:spot_meter_mode":"W","Z3RequestHandler:post:CameraControl:spot_meter_pos_inquiry":"R","Z3RequestHandler:post:CameraControl:stable_zoom_inquiry":"R","Z3RequestHandler:post:CameraControl:stable_zoom_off":"W","Z3RequestHandler:post:CameraControl:stable_zoom_on":"W","Z3RequestHandler:post:CameraControl:tnr_disable":"W","Z3RequestHandler:post:CameraControl:tnr_enable":"W","Z3RequestHandler:post:CameraControl:version":"R","Z3RequestHandler:post:CameraControl:version_inquiry":"R","Z3RequestHandler:post:CameraControl:visibility_enhancer_inquiry":"R","Z3RequestHandler:post:CameraControl:visibility_enhancer_off":"W","Z3RequestHandler:post:CameraControl:visibility_enhancer_on":"W","Z3RequestHandler:post:CameraControl:visibility_enhancer_params":"W","Z3RequestHandler:post:CameraControl:visibility_enhancer_params_inquiry":"W","Z3RequestHandler:post:CameraControl:wb_auto_mode":"W","Z3RequestHandler:post:CameraControl:wb_auto_trace_mode":"W","Z3RequestHandler:post:CameraControl:wb_autotrace_mode":"W","Z3RequestHandler:post:CameraControl:wb_get_bgain":"R","Z3RequestHandler:post:CameraControl:wb_get_mode":"R","Z3RequestHandler:post:CameraControl:wb_get_mode_name":"R","Z3RequestHandler:post:CameraControl:wb_get_rgain":"R","Z3RequestHandler:post:CameraControl:wb_indoor_mode":"W","Z3RequestHandler:post:CameraControl:wb_manual_bgain_direct":"W","Z3RequestHandler:post:CameraControl:wb_manual_bgain_reset":"W","Z3RequestHandler:post:CameraControl:wb_manual_mode":"W","Z3RequestHandler:post:CameraControl:wb_manual_rgain_direct":"W","Z3RequestHandler:post:CameraControl:wb_manual_rgain_reset":"W","Z3RequestHandler:post:CameraControl:wb_onepush_mode":"W","Z3RequestHan

dler:post:CameraControl:wb_onepush_trigger":"W","Z3RequestHandler:post:Camer
aControl:wb_outdoor_auto_mode":"W","Z3RequestHandler:post:CameraControl:wb_o
utdoor_mode":"W","Z3RequestHandler:post:CameraControl:wb_sodium_lamp_auto_mo
de":"W","Z3RequestHandler:post:CameraControl:wb_sodium_lamp_fixed_mode":"W",
"Z3RequestHandler:post:CameraControl:wb_sodium_lamp_outdoor_mode":"W","Z3Req
uestHandler:post:CameraControl:white_hot":"W","Z3RequestHandler:post:CameraC
ontrol:wide_dynamic_range_inquiry":"R","Z3RequestHandler:post:CameraControl:
wide_dynamic_range_off":"W","Z3RequestHandler:post:CameraControl:wide_dynami
c_range_on":"W","Z3RequestHandler:post:CameraControl:zoom_direct":"W","Z3Req
uestHandler:post:CameraControl:zoom_get_focal_length_mm":"R","Z3RequestHandl
er:post:CameraControl:zoom_get_pos":"R","Z3RequestHandler:post:CameraControl
:zoom_stop":"W","Z3RequestHandler:post:CameraControl:zoom_tele_std":"W","Z3R
equestHandler:post:CameraControl:zoom_tele_var":"W","Z3RequestHandler:post:C
ameraControl:zoom_wide_std":"W","Z3RequestHandler:post:CameraControl:zoom_wi
de_var":"W","Z3RequestHandler:post:DecoderStatus":"?","Z3RequestHandler:post
:DeleteChannel":"W","Z3RequestHandler:post:DeletePreset":"W","Z3RequestHandl
er:post:DisplayStatus":"?","Z3RequestHandler:post:Dynamic:analog_gain_db":"?
","Z3RequestHandler:post:Dynamic:crop":"?","Z3RequestHandler:post:Dynamic:go
p":"?","Z3RequestHandler:post:Dynamic:nfstrength":"?","Z3RequestHandler:post
:Dynamic:pip_enable":"?","Z3RequestHandler:post:Dynamic:pip_location":"?","Z
3RequestHandler:post:Dynamic:rotate_angle":"?","Z3RequestHandler:post:Dynami
c:rotate_center_offset":"?","Z3RequestHandler:post:Dynamic:rotate_enable":"?
","Z3RequestHandler:post:Dynamic:rotate_mode":"?","Z3RequestHandler:post:Dyn
amic:startmulticast":"?","Z3RequestHandler:post:Dynamic:stopmulticast":"?","
Z3RequestHandler:post:Dynamic:telop_text":"?","Z3RequestHandler:post:Dynamic
:translate_enable":"?","Z3RequestHandler:post:Dynamic:translate_offset":"?",
"Z3RequestHandler:post:Dynamic:vconflict":"?","Z3RequestHandler:post:Dynamic
:vrate":"?","Z3RequestHandler:post:Dynamic:vratediv":"?","Z3RequestHandler:p
ost:EjectStorage":"W","Z3RequestHandler:post:EncoderStatus":"?","Z3RequestHa
ndler:post:ErasePresets":"W","Z3RequestHandler:post:ExportPresets":"W","Z3Re
questHandler:post:FactoryReset":"W","Z3RequestHandler:post:GetCameraLink":"?
","Z3RequestHandler:post:GetCameraTab":"?","Z3RequestHandler:post:GetStatus"
:"?","Z3RequestHandler:post:GetVideoGroup":"?","Z3RequestHandler:post:GetWif
iPass":"?","Z3RequestHandler:post:InsQuery":"W","Z3RequestHandler:post:LoadU
ser":"W","Z3RequestHandler:post:Login":"W","Z3RequestHandler:post:Logout":"W
","Z3RequestHandler:post:Overlay":"?","Z3RequestHandler:post:OverlayStop":"W
","Z3RequestHandler:post:PtzAbsoluteMove":"W","Z3RequestHandler:post:PtzAuxi
liary":"W","Z3RequestHandler:post:PtzClearPreset":"W","Z3RequestHandler:post
:PtzContinuousMove":"W","Z3RequestHandler:post:PtzGotoPreset":"W","Z3Request
Handler:post:PtzNewTourSpot":"W","Z3RequestHandler:post:PtzPosition":"W","Z3
RequestHandler:post:PtzPreset":"W","Z3RequestHandler:post:PtzQuery":"W","Z3R
equestHandler:post:PtzRemoveTourSpot":"W","Z3RequestHandler:post:PtzSetPrese
t":"W","Z3RequestHandler:post:PtzStartTour":"W","Z3RequestHandler:post:PtzSt
op":"W","Z3RequestHandler:post:RemoveJobs":"W","Z3RequestHandler:post:Restar
tBoard":"W","Z3RequestHandler:post:SaveCamera":"?","Z3RequestHandler:post:Sa
veCameraBoson":"?","Z3RequestHandler:post:SaveCameraDRSTamarisk":"?","Z3Requ
estHandler:post:SaveCameraLink":"?","Z3RequestHandler:post:SaveCameraLx":"?"
,"Z3RequestHandler:post:SaveCameraTau":"?","Z3RequestHandler:post:SaveJobs":
"W","Z3RequestHandler:post:SavePresets":"W","Z3RequestHandler:post:SaveUser"
:"W","Z3RequestHandler:post:SetAPConfig":"?","Z3RequestHandler:post:SetAdv":
"?","Z3RequestHandler:post:SetAudio":"W","Z3RequestHandler:post:SetCameraLin

k":"W","Z3RequestHandler:post:SetConsole":"?","Z3RequestHandler:post:SetDDNS
Enable":"W","Z3RequestHandler:post:SetDSCP":"?","Z3RequestHandler:post:SetDe
bugLevel":"W","Z3RequestHandler:post:SetDevName":"W","Z3RequestHandler:post:
SetDisplay":"?","Z3RequestHandler:post:SetEncoder":"W","Z3RequestHandler:pos
t:SetEncoderCameraControl":"W","Z3RequestHandler:post:SetFpga":"W","Z3Reques
tHandler:post:SetIp":"W","Z3RequestHandler:post:SetLoginLimit":"W","Z3Reques
tHandler:post:SetNFS":"W","Z3RequestHandler:post:SetNMEAEnable":"?","Z3Reque
stHandler:post:SetOnvif":"W","Z3RequestHandler:post:SetOnvifVMD":"W","Z3Requ
estHandler:post:SetRMF":"W","Z3RequestHandler:post:SetRTSP":"W","Z3RequestHa
ndler:post:SetSNTP":"W","Z3RequestHandler:post:SetStartLogs":"W","Z3RequestH
andler:post:SetStopLogs":"W","Z3RequestHandler:post:SetTermSrvEnable":"?","Z
3RequestHandler:post:SetViVpssMode":"?","Z3RequestHandler:post:SetVideoGroup
":"W","Z3RequestHandler:post:SetViewport":"?","Z3RequestHandler:post:SetWifi
AP":"?","Z3RequestHandler:post:SetZFinderEnable":"W","Z3RequestHandler:post:
SourceStatus":"?","Z3RequestHandler:post:StartChannel":"W","Z3RequestHandler
:post:StartMTS":"W","Z3RequestHandler:post:StopChannel":"W","Z3RequestHandle
r:post:StopMTS":"W","Z3RequestHandler:post:StreamStatus":"?","Z3RequestHandl
er:post:TempStatus":"?","Z3RequestHandler:post:UpdatePtz":"?","Z3RequestHand
ler:post:UpdatePtzPreset":"W","Z3RequestHandler:post:UpdatePtzTourSpots":"W"
,"Z3RequestHandler:post:UpdateTerm":"W","Z3RequestHandler:post:UpdateTermFor
Fpga":"W","Z3RequestHandler:post:ViscaPort":"x","Z3RequestHandler:post:camer
a_inquiry":"?","Z3RequestHandler:post:fmt_media":"W","Z3RequestHandler:post:
update_require_web_login":"W","Z3RequestHandler:post:user_add":"W","Z3Reques
tHandler:post:user_remove":"W","Z3RequestHandler:post:user_update_level":"W"
,"Z3RequestHandler:post:user_update_password":"W","Z3SSLImportHandler:post":
"W","Z3SnapshotHandler:get":"R","Z3TabHandler:get":"-
","Z3UploadHandler:post":"W","Z3WebSocketHandler:get":"R"},"ret":"0","status
":"OK","userlevel":0,"username":"admin"}

---

## ptz

---

Get the current PTZ settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***token*** TEXT,

***node*** TEXT.

***name*** TEXT.

***enabled*** BOOLEAN.

***pelcoaddr*** INTEGER.

***tcpport*** INTEGER.

***ptztype*** TEXT.

***flip*** TEXT.

***pos*** TEXT.

***maxtilt*** INTEGER.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
PTZ response:
{"data":[{"id":1,"values":{"db_videosource":"CAMERA","enabled":1,"flip":"non
e","maxtilt":90,"name":"videoin1","pelcoaddr":1,"pos":"simulated","ptztype":
"relative","rowid":1,"tcpport":2000,"token":"ptz"}},{"id":2,"values":{"db_vi
deosource":"CAMERA2","enabled":1,"flip":"none","maxtilt":90,"name":"videoin2
","pelcoaddr":2,"pos":"simulated","ptztype":"relative","rowid":2,"tcpport":2
000,"token":"ptz2"}}],"metadata":[{"datatype":"text","editable":false,"label
":"Video
Source","name":"db_videosource"},{"datatype":"text","editable":false,"label"
:"ONVIF
Token","name":"token"},{"datatype":"text","editable":true,"label":"ONVIF
Name","name":"name"},{"datatype":"bool","editable":true,"label":"Enabled","n
ame":"enabled"},{"datatype":"integer","editable":true,"label":"RS485
Address","name":"pelcoaddr"},{"datatype":"integer","editable":true,"label":"
TCP
Port","name":"tcpport"},{"datatype":"text","editable":true,"label":"Type","n
ame":"ptztype"},{"datatype":"text","editable":true,"label":"Flip","name":"fl
ip"},{"datatype":"text","editable":true,"label":"Position","name":"pos"},{"d
atatype":"integer","editable":true,"label":"Max
Tilt","name":"maxtilt"}],"ret":"0"}
```

## ptz_preset

Get the PTZ preseting values from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***profiletoken*** TEXT,

***token*** TEXT.

***name*** TEXT.

***pan*** FLOAT.

***tilt*** FLOAT.

***zoom*** FLOAT.

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
PTZ Preset response:
{"data":[{"id":1,"values":{"action":"blank","name":"home","profiletoken":"pr
ofile1","rowid":1,"token":"1"}}],"metadata":[{"datatype":"text","editable":f
alse,"label":"Profile","name":"profiletoken"},{"datatype":"text","editable":
false,"label":"PresetID","name":"token"},{"datatype":"text","editable":true,
"label":"Name","name":"name"},{"datatype":"html","editable":false,"label":"A
ction","name":"action"}],"ret":0}
```

## ptz_tour

Get the PTZ tour information from the Database.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***ProfileToken*** TEXT,

***PresetTourToken*** TEXT.

***name*** TEXT.

***autostart*** BOOLEAN.

***RecurringTime*** INTEGER.

***RecurringDuration*** TEXT.

***PresetTourDirection*** INTEGER.

***RandomPresetOrder*** BOOLEAN.

***ProfileToken*** TEXT.

***PresetTourToken*** TEXT.

***RowIndex*** INTEGER.

***PresetDetailToken*** TEXT.

***PresetDetailHome*** BOOLEAN.

***StayTime*** TEXT.

***pan*** FLOAT.

***tilt*** FLOAT.

***zoom*** FLOAT.

***pspeed*** FLOAT.

***tspeed*** FLOAT.

***zspeed*** FLOAT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
PTZ tour response:
{"preset":{"data":[{"id":1,"values":{"name":"home","pan":0,"profiletoken":"p
rofile1","rowid":1,"tilt":0,"token":"1","zoom":0}}],"metadata":[{"datatype":
"text","editable":true,"label":"Profiletoken","name":"profiletoken"},{"datat
ype":"text","editable":true,"label":"Token","name":"token"},{"datatype":"tex
t","editable":true,"label":"Name","name":"name"},{"datatype":"real","editabl
e":true,"label":"Pan","name":"pan"},{"datatype":"real","editable":true,"labe
l":"Tilt","name":"tilt"},{"datatype":"real","editable":true,"label":"Zoom","
name":"zoom"}]},"ret":0,"spot":{"data":[],"metadata":[{"datatype":"text","ed
itable":false,"label":"Profile","name":"ProfileToken"},{"datatype":"text","e
ditable":false,"label":"Tour","name":"PresetTourToken"},{"datatype":"text","
editable":true,"label":"Preset","name":"PresetDetailToken"},{"datatype":"tex
t","editable":true,"label":"StayTime","name":"StayTime"},{"datatype":"intege
r","editable":false,"label":"RowIndex","name":"RowIndex"},{"datatype":"html"
,"editable":"false","label":"Action","name":"action"}]},"tour":{"data":[{"id
":1,"values":{"PresetTourToken":"1","ProfileToken":"profile1","name":"tour1"
,"rowid":1}},{"id":2,"values":{"PresetTourToken":"2","ProfileToken":"profile
1","name":"tour2","rowid":2}},{"id":3,"values":{"PresetTourToken":"3","Profi
leToken":"profile1","name":"tour3","rowid":3}},{"id":4,"values":{"PresetTour
Token":"4","ProfileToken":"profile1","name":"tour4","rowid":4}},{"id":5,"val
ues":{"PresetTourToken":"5","ProfileToken":"profile1","name":"tour5","rowid"
:5}},{"id":6,"values":{"PresetTourToken":"6","ProfileToken":"profile1","name
":"tour6","rowid":6}},{"id":7,"values":{"PresetTourToken":"7","ProfileToken"
:"profile1","name":"tour7","rowid":7}},{"id":8,"values":{"PresetTourToken":"
8","ProfileToken":"profile1","name":"tour8","rowid":8}},{"id":9,"values":{"P
resetTourToken":"9","ProfileToken":"profile1","name":"tour9","rowid":9}},{"i
d":10,"values":{"PresetTourToken":"10","ProfileToken":"profile1","name":"tou
r10","rowid":10}},{"id":11,"values":{"PresetTourToken":"11","ProfileToken":"
profile1","name":"tour11","rowid":11}},{"id":12,"values":{"PresetTourToken":
"12","ProfileToken":"profile1","name":"tour12","rowid":12}},{"id":13,"values
":{"PresetTourToken":"13","ProfileToken":"profile1","name":"tour13","rowid":
```

```
13}},{"id":14,"values":{"PresetTourToken":"14","ProfileToken":"profile1","na
me":"tour14","rowid":14}},{"id":15,"values":{"PresetTourToken":"15","Profile
Token":"profile1","name":"tour15","rowid":15}},{"id":16,"values":{"PresetTou
rToken":"16","ProfileToken":"profile1","name":"tour16","rowid":16}}],"metada
ta":[{"datatype":"text","editable":false,"label":"profile1","name":"ProfileT
oken"},{"datatype":"text","editable":false,"label":"TourID","name":"PresetTo
urToken"},{"datatype":"text","editable":true,"label":"Name","name":"name"}]}
}
```


---

## snmp

---

Get the Simple Network Management Protocol values from the Database

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***power_trap_en*** INTEGER,

***input_loss_trap_en*** INTEGER.

***input_recover_trap_en*** INTEGER.

***temp_high_trap_en*** INTEGER.

***temp_recover_trap_en*** INTEGER.

***high_temp*** INTEGER.

***nominal_temp*** INTEGER.

***high_temp_suppress*** INTEGER.

***nominal_temp_suppress*** INTEGER.

***trap_hosts*** INTEGER.

***snmp_v3_en*** INTEGER.

***snmp_v3_user*** TEXT.

***snmp_v3_passwd*** TEXT.

***snmp_v3_usrpro*** TEXT.

***snmp_v3_encrypt*** INTEGER.

***snmp_v3_encrypt_passwd*** TEXT.

***snmp_v3_encpro*** TEXT.

***snmp_v3_engineid*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
SNMP response:
{"high_temp":80,"high_temp_suppress":30,"input_loss_trap_en":0,"input_recove
r_trap_en":0,"nominal_temp":60,"nominal_temp_suppress":30,"power_trap_en":0,
"ret":"0","snmp_v3_en":0,"snmp_v3_encpro":"des","snmp_v3_encrypt":0,"snmp_v3
_encrypt_passwd":"z3password","snmp_v3_engineid":"","snmp_v3_passwd":"z3pass
word","snmp_v3_user":"z3user","snmp_v3_usrpro":"md5","temp_high_trap_en":0,"
temp_recover_trap_en":0,"trap_hosts":"192.168.0.6"}
```

---

## ssl

---

Get the Secure Socket Layer settings from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***https_enable*** TEXT,

***http_enable*** TEXT.

***https_port*** INTEGER.

***http_port*** INTEGER.

***cert_status*** TEXT.

***key_status*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
SSL response:
{"cert_status":"Unavailable","http_enable":"on","http_port":80,"https_enable
":"off","https_port":443,"key_status":"Unavailable","ret":"0"}
```

---

## stats

---

Get the available memory infromation from the device.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***mem_free_kb*** TEXT,

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
STATUS response: {"astream_status_str":"\tChannel 1 Codec fdk_aaclc
Samplerate 48000 Input MIC Frames 546732 \tChannel 11 Codec fdk_aaclc
Samplerate 48000 Input MIC Frames 545860 +OK","encoder_status_str":"   ***
Encode Bitstream Received Statistics ***    CH | Bitrate (Kbps) | Actual
Bitrate | FPS  | Actual FPS | Key-frame FPS | Width | Height   -------------
-------------------------------------------------------------------------
------------------   1 |        6000.00 |        6006.33 | 29.97 |    29.97
|           0.5 |  3840 |   2160  |
+OK","mem_free_kb":670856,"ret":"0","source_status_str":"+CAMERA 3840x2160p
29.97 fps\n","status":"OK","stream_status_str":"Channel 1 Codec H265-(HEVC)
URL rtsp Frames 348194 +OK","temp_status_str":"+LENS 29 +CPU 38.0 +OK"}
```

## sys

Get the current system settings from the Database and the ip information from the device.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***local_ip*** TEXT,

***local_netmask*** TEXT.

***preset*** TEXT.

***enc_channels*** TEXT.

***opmode*** TEXT.

***username*** TEXT.

***userlevel*** TEXT.

***do_autostart*** BOOLEAN.

***syspassword*** TEXT.

***session_id*** TEXT.

***disp_std*** TEXT.

***disp_std2*** TEXT.

***disp_input*** TEXT.

***disp_mode*** TEXT.

***disp_layout*** TEXT.

***enc_adv_setting*** TEXT.

***enc_vivpss_mode_enable*** TEXT.

***enable_sntp*** TEXT.

***sntp_servers*** TEXT.

***enable_snmp*** TEXT.

***sysdevicename*** TEXT.

***timezone*** TEXT.

***timezone_name*** TEXT.

***termserve_remote_enable*** TEXT.

***termserve_shared_enable*** TEXT.

***zfinder_enable*** TEXT.

***diff_serve*** INTEGER.

***enable_ptp*** TEXT.

***ptp_role*** TEXT.

***nmea_enable*** TEXT.

***nfs_enable*** TEXT.

***nfs_server*** TEXT.

***nfs_server_root*** TEXT.

***ddns_enable*** TEXT.

***ddns_provider*** TEXT.

***ddns_username*** TEXT.

***ddns_password*** TEXT.

***rtspd_port*** INTEGER.

***rtspd_timeout*** INTEGER.

***maxlogin*** INTEGER.

***login_window*** INTEGER.

***lockout_interval*** INTEGER.

***login_timeout*** INTEGER.

***require_web_login*** BOOLEAN.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
SYSTEM response:
{"ANALOG_analog_gain_db":7,"MICL_analog_gain_db":-12,"MIC_analog_gain_db":7,
"console_enable":"on","ddns_enable":"off","ddns_hostname":"","ddns_password"
:"","ddns_provider":"freedns","ddns_username":"","diff_serve":0,"disp_input"
:"primary","disp_layout":"1x1","disp_mode":"ultrahd","disp_std":"auto","disp
_std2":"auto","do_autostart":1,"enable_ptp":"false","enable_snmp":"false","e
nable_sntp":"true","enc_adv_setting":"off","enc_vivpss_mode_enable":"off","l
ocal_ip":"192.168.10.127","local_netmask":"255.255.255.0","lockout_interval"
:900,"login_timeout":900,"login_window":900,"maxlogin":3,"nfs_enable":"off",
"nfs_server":"192.168.1.6","nfs_server_root":"/c/media","nmea_enable":"off",
"pe0":"encoder","pe1":"","pe2":"","pe3":"","pe4":"","pe5":"","pe6":"","pe7":
"","pe8":"","ptp_role":"auto","require_web_login":0,"ret":"0","rtspd_port":5
54,"rtspd_timeout":60,"session_id":"---
","sntp_servers":"pool.ntp.org","sysdevicename":"Z3Cam","syspassword":"","te
```

```
rmserve_remote_enable":"on","termserve_shared_enable":"on","timezone":"CST6C
DT,M3.2.0,M11.1.0","timezone_name":"America/Chicago","userlevel":0,"username
":"admin","wifi_exists":false,"zfinder_enable":"on"}
```


---

## term

---

Get the remote terminal information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***term_mode*** TEXT,

***term_protocol*** TEXT.

***term_localport*** TEXT.

***term_servaddr*** TEXT.

***term_servport*** TEXT.

***term_data_bits*** TEXT.

***term_parity*** TEXT.

***term_stop_bits*** TEXT.

***term_baudrate*** TEXT.

***term_devicefile*** TEXT.

***term_function*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
```

```
admin
```

---

**Example output:**

```
TERM response:
{"data":[{"id":1,"values":{"rowid":1,"term_baudrate":"2400","term_data_bits"
:"8","term_devicefile":"/dev/ttyS4","term_function":"rs485","term_localport"
:"2000","term_parity":"None","term_stop_bits":"1"}},{"id":2,"values":{"rowid
":2,"term_baudrate":"9600","term_data_bits":"8","term_devicefile":"/dev/ttyS
2","term_function":"camera1_visca","term_localport":"1000","term_parity":"No
ne","term_stop_bits":"1"}},{"id":3,"values":{"rowid":3,"term_baudrate":"9600
","term_data_bits":"8","term_devicefile":"/dev/ttyS3","term_function":"off",
"term_localport":"1001","term_parity":"None","term_stop_bits":"1"}},{"id":4,
"values":{"rowid":4,"term_baudrate":"115200","term_data_bits":"8","term_devi
cefile":"/dev/ttyS1","term_function":"user","term_localport":"1500","term_pa
rity":"None","term_stop_bits":"1"}},{"id":5,"values":{"rowid":5,"term_baudra
te":"115200","term_data_bits":"8","term_devicefile":"/dev/ttyS0","term_funct
ion":"console","term_localport":"1800","term_parity":"None","term_stop_bits"
:"1"}}],"metadata":[{"datatype":"text","editable":true,"label":"Serial
Port","name":"term_devicefile"},{"datatype":"text","editable":true,"label":"
Baud
Rate","name":"term_baudrate"},{"datatype":"text","editable":true,"label":"Da
ta
Bits","name":"term_data_bits"},{"datatype":"text","editable":true,"label":"P
arity","name":"term_parity"},{"datatype":"text","editable":true,"label":"Sto
p
Bits","name":"term_stop_bits"},{"datatype":"text","editable":true,"label":"F
unction","name":"term_function"},{"datatype":"text","editable":true,"label":
"TCP Port","name":"term_localport"}],"ret":"0"}
```

---

## users

---

Get the ONVIF user information and the permission level from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***Name*** TEXT,

***Level*** INTEGER.

***Password*** TEXT.

***Password_MD5*** TEXT.

***Password_SHA256*** TEXT.

***Password_SHA512_256*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
USERS response:
{"data":[{"id":1,"values":{"Level":"0","Name":"admin","action":"blank","rowi
d":1}},{"id":2,"values":{"Level":"7","Name":"guest","action":"blank","rowid"
:2}}],"metadata":[{"datatype":"text","editable":false,"label":"Name","name":
"Name"},{"datatype":"text","editable":true,"label":"Level","name":"Level"},{
"datatype":"html","editable":false,"label":"Action","name":"action"}],"ret":
0}
```

---

## video_group

---

Get the current video group settings from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***vgroup*** INTEGER.

***crop_enable*** TEXT.

***crop_width*** INTEGER.

***crop_height*** INTEGER.

***crop_x*** INTEGER.

***crop_y*** INTEGER.

***nr_enable*** TEXT.

---

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

---

**Example output:**

```
Video Group response:
{"crop_enable":"off","crop_height":0,"crop_width":0,"crop_x":0,"crop_y":0,"n
r_enable":"off","ret":"0","vgroup":1}
```

---

## wifi

---

Get the Wifi details from the device and wifi client information from the Database.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***networks*** TEXT,

***cur_network*** TEXT,

***cur_pass*** TEXT,

***ap_pass*** TEXT,

***ap_pass_en*** BOOLEAN.

***ap_ssid*** TEXT,

***wifi_client_local_ip*** TEXT,

***wifi_client_local_netmask*** TEXT.

***wifi_client_default_gw*** TEXT.

***wifi_client_use_dhcp*** BOOLEAN.

---

## zoomStepSize

---

Get the current camera zoom step values from the Database.

**Parameters:**

---

***chn*** Get the channel number. The default is Camera 1. Select between connected cameras. Possible values: 1 = Camera 1, 2 = Camera 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***zoom_step_size*** INTEGER,

**Command to execute:**

```
python3 status_DIGEST_ARGPARSE.py -ip 192.168.10.127 -username admin -passwd
admin
```

**Example output:**

```
Zoom Step Size response: {"ret":"0","size":1}
```

# File Upload for MFR or UPD Image Control

## Upload Image (POST)

Upload the z3 image for e.g. (z3-30-0139-mfr.img or z3-30-0139-upd.img) in the z3 device.

Before uploading the new image, it checks, whether the device has enough free memory or not and removes the coredump files. If the image is *mfr.img then it removes the default database from the data partition. If its *upd.img then it will not remove the database values.

Note:
It takes approximately 5 minutes to update; DO NOT SEND ANY OTHER API Commands after uploading the image for at least 5 minutes. (time.sleep(300) or equivalent in scripts)

**Parameters:**

***img_file*** Get the file name which needs to be flashed in the z3 device.

**Returns:**

Flash the image and reboot the device.

Note:
It takes approximately 5 minutes to update; DO NOT SEND ANY OTHER API Commands after uploading the image for at least 5 minutes. (time.sleep(300) or equivalent in scripts)

# Download File and Remove File Control

See **download_remove** to remove files from the **Generic Actions (POST)** action

## Download Media (GET)

```
http://<ENCODER_IP>/download_media.cgi?mediafile=/location/filename
```

For e.g.

```
http://<ENCODER_IP>/download_media.cgi?mediafile=/media/sda1/MOV1_000001.mp4
```

---

Download the media file. If the output format is mp4 or TS file, then only z3 device allows downloading the files. Also provides the option for deleting the file.

**Parameters:**

---

***mediafile*** Get the media file name.

**Returns:**

---

JSON formatted table with values below.

ret Custom return code of the HTTP POST request. 0 on success, -1 on failure.

***File Name*** Name of the download file name .

***File Size*** Size of the download file name.

***Last Modified*** Last modified the file date and time.

---

## download.html to get list of downloadable files

```
http://<ENCODER_IP>/download.html?chn=1
```

chn **=[1,2,3,4]**

List the download content. The content will be either .mp4 or .ts file format.

# Encoder Actions (POST)

### SetEncoder

Write encoder settings to active preset. See Python example code for Seting Configuration below

### EncoderStatus

Requests an update to encoder_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

### StreamStatus

Requests an update to stream_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

### AStreamStatus

Requests an update to astream_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

### SourceStatus

Requests an update to source_status_str variable.

To read the result, you need to POST the “stats” control as shown\ in the Python example code below.

---

# Setting Encoder Configuration

To set the encode configuration per channel the **SetEncoder** action is used with a post. All variables should be present or they will be replaced with a defualt that may not be valid for your application. More than one channel can be set per command. It is recommended you first read the current settings from the encoder then modify them and send the new settings back as in the Python

example below:

```
import requests, json, sys

server_url='http://192.168.0.120/cgi-bin/control.cgi'
channel = 1

enc_cfg = requests.get(server_url, params='ctrl=enc&chn={}'.format(channel))
print enc_cfg.json()

enc_cfg_json=enc_cfg.json()
enc_cfg_json['vframeratediv'] = 2 #set frame rate divider to 2
new_cfg = {}
new_cfg['action'] = 'SetEncoder'
for key, value in enc_cfg_json.iteritems():
print "key = {} value = {}".format(key,value)
new_idx = "enc_{}_{}".format(channel,key)
print "setting {} to {}".format(new_idx,value)
new_cfg[new_idx] = value

print new_cfg
requests.post(server_url, data=new_cfg)
```

# snapshot.cgi Control

```
http://<ENCODER_IP>/snapshot.cgi?size=3840x2160&quality=100

http://<ENCODER_IP>/snapshot.cgi?chn=1&size=1280x720&quality=80
```

valid options are:

quality **=[20-100]**

* *size **=[any supported valid resolution]**

chn **=[1,2,3,4]**

```
quality** =[0-100] supported resolutions are: 4K Models Only 3840×2160
2560×1440 4K and HD Models 1920×1080
1440×1080
1280×1024
1280×720
1024×768
960×720
1024×576
800×600
720×576
704×576
720×480
```

```
640×512
640×480
640×360
352×576
420×380
352×288
352×240
336×256
320×240
320×180 **
```

# Reading Statistics

**Python example of updating and reading encoder statistics:**

```
import requests, json, sys
server_url='http://192.168.0.120/cgi-bin/control.cgi'

requests.post(server_url, {'action': 'EncoderStatus'} )
requests.post(server_url, {'action': 'StreamStatus'}  )
requests.post(server_url, {'action': 'AStreamStatus'} )
requests.post(server_url, {'action': 'SourceStatus'}  )
requests.post(server_url, {'action': 'TempStatus'}  )

print requests.get(server_url, params='ctrl=stats&chn=null').json()
```

**Example output:**

```
{u'status': u'OK',
u'astream_status_str': u'\tChannel 6 Codec fdk_aaclc Samplerate 48000 Input
MICL Frames 32287820  OK',
u'temp_status_str': u' LENS 29  CPU 78.700  FPGA 89.8  OK',
u'ret': u'0',
u'encoder_status_str': u'   *** Encode Bitstream Received Statistics ***
CH | Bitrate (Kbps) | Actual Bitrate | FPS  | Actual FPS | Key-frame FPS |
Width | Height   ------------------------------------------------------------
-----------------------------------------------
1 |        4000.00 |        4092.18 | 60.0 |     59.8 |           1.0 |
1920 |   1080  |  OK',
u'source_status_str': u' CAMERA 1920x1080p 60.00 fps\n',
u'stream_status_str': u'Channel 1 URL rtsp Frames 12883581  OK'}
```

# Authentication

By default, no authentication is required to access the HTTP API.

To enable authentication, go to the “System” tab “Device Management” section. Click on the “Set

Password” button.

The username will be “admin” The authentication method is HTTP Digest authentication. HTTP Basic authentication is not supported.

#### Python example code of HTTP Digest authentication

```
import requests,json, sys
from requests.auth import HTTPDigestAuth

auth=HTTPDigestAuth('admin','mypassword')

print requests.get(server_url, params='ctrl=stats&chn=null',
auth=auth).json()
```

# Appendix A: Sample Python Scripts

The sample python scripts below require Python3.x.

The requests package must be installed.

You can install using

1) Use the Ubuntu package manager

sudo apt install python3-requests

2) Use the Python3 native package installer

sudo pip3 install requests

## SaveUser (saveUser_DIGEST_ARGPARSE.py)

Save the state, preset, encoder/decoder settings for the user.

#### Python example code of HTTP Digest SaveUser

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth
```

```
def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
payload = "action=SaveUser&enc_current_preset=charles";
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   setip_respose response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
```

```
  )
)
```

**Command to execute**

```
python3 saveUser_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -
passwd admin
```

**Example output:**

```
SaveUser response: {"ret":"0","status":"OK"}
```

# SetIp (set_ip_DIGEST_ARGPARSE.py)

Control networking settings of the Z3 device such as local IPv4 IP address, netmask, DNS Server primary, DNS Server secondary, etc. Additionally control Encoder Auto-Start after boot-up and enable/disable showing of advanced WebUI settings.

**Python example code of HTTP Digest SetIp**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
```

```
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-newip', help='Set this IP')
parser.add_argument('-mask', help='Subnet Mask',default='255.255.255.0')
parser.add_argument('-gateway', help='Network Gateway',default='172.30.1.1')
parser.add_argument('-dns1', help='DNS 1',default='8.8.8.8')
parser.add_argument('-dns2', help='DNS 2',default='8.8.4.4')
parser.add_argument('-ipmtu', help='MTU Speed',default=1500)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
payload =
"action=SetIp&local_ip=172.30.1.112&local_netmask=255.255.255.0&default_gw=1
72.30.1.1&local_dnsip=8.8.8.8&local_dnsip2=8.8.4.4&use_dhcp=no&ipmtu=1500ð_s
peed=AUTOð_duplex=AUTO&do_autostart=1&enc_adv_setting=off";
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   setip_respose response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 set_ip_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -passwd
admin
```

**Example output:**

```
dns1='8.8.8.8', dns2='8.8.4.4', gateway='172.30.1.1', ip='192.168.1.105',
ipmtu=1500, mask='255.255.255.0', newip=None, passwd='admin',
username='admin'
setip_respose response: {"ret":"0","status":"OK"}
```

# cam_state (sys_DIGEST_ARGPARSE.py)

Get the current camera state information from the Database.

### Python example code of HTTP Digest camera state

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
from requests.auth import HTTPDigestAuth

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    print(msg, flush=True)

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
```

```
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)

if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)

control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}

payload='ctrl=cam_state&chn=null';
rs = http_request_get(payload)
print_immediately(
  "{:>6}: {}   Sys response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 sys_DIGEST_ARGPARSE.py -ip 192.168.1.105 -username admin -passwd
admin
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin')
Sys response:
{"camera1_if_type":"Sony_LVDS","camera2_if_type":"CVBS","enc_channels":"1,2"
,"enc_current_preset":"charles","fpgafileglob":"fpga_none.bin|fpga2_0139_cvb
s.bin","preview_auto_start":"1","ret":"0","visca_present":"true"}
```

# CameraControl for Boson camera (Zoom_Boson.py)

Send control commands to boson camera. Only one command can be sent at a time;

#### Python example code of HTTP Digest Camera control for Boson camera

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
import sys
from requests.auth import HTTPDigestAuth

if sys.version_info[0] <3:
  raise Exception("Python 3 or a more recent version is required.")

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    # print(msg, flush=True)
    print(msg)
    sys.stdout.flush()

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
```

```
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
args = parser.parse_args()
print(args)
if args.ip:
  server_url='http://' args.ip
elif args.ip_address:
  server_url='http://' args.ip_address
# Set Zoom STEP
zooms=args.zs;
zoomstep=args.zoom_step;

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)
control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
# Boson min Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000000000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
```

```
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson mid Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000150000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson mid Zoom xB0 yA0
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000150000000B0000000A0';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
```

```
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson max Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000030000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
# Boson no Zoom
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000d000200000000000000A000000080';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
   datetime.datetime.now(),
   rs.text
  )
)
time.sleep(4)
# Boson get Zoom pos
payload='action=CameraControl&interface=Boson&cam_index=2&command=raw
000D0003';
rs = http_request(payload)
print_immediately(
  "{:>6}: {}   Response: {}".format(
   1,
```

```
   datetime.datetime.now(),
   rs.text
  )
)
```

**Command to execute**

```
python3 Zoom_Boson.py -ip 172.28.30.103
```

**Example output:**

```
(ip='172.28.30.103', passwd='admin', username=None, zoom_step=False, zs='3')
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000000000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000015000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000015000000B0000000A0","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
Response: {"response":"00000030000000A000000080","ret":"0","status":"OK"}
Response: {"response":"","ret":"0","status":"OK"}
```

# CameraControl for Sony Visca camera (Zoom_visca.py)

Send control commands to visca camera. Only one command can be sent at a time;

**Python example code of HTTP Digest Camera control for Visca camera**

```
import requests
import json
import sched
import time
import enum
import datetime
import argparse
import sys
from requests.auth import HTTPDigestAuth

if sys.version_info[0] <3:
  raise Exception("Python 3 or a more recent version is required.")

def print_immediately(msg):
    #
https://stackoverflow.com/questions/230751/how-can-i-flush-the-output-of-the
-print-function/230774#230774
    # print(msg, flush=True)
    print(msg)
```

```
    sys.stdout.flush()

def http_request(payload):
  try:
    if args.username:
      r = requests.post(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.post(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

def http_request_get(payload):
  try:
    if args.username:
      r = requests.get(control_cgi_url, data=payload,
headers=headers,auth=auth)
    else:
      r = requests.get(control_cgi_url, data=payload, headers=headers)
  except requests.exceptions.HTTPError as err:
   raise SystemExit(err)
  except requests.exceptions.Timeout:
   raise SystemExit()
  except requests.exceptions.TooManyRedirects:
   raise SystemExit()
  except requests.exceptions.RequestException as e:
   raise SystemExit(e)
  return r

parser = argparse.ArgumentParser()
parser.add_argument('-ip', '-ip_address', type=str, help='IP of Encoder',
required=True)
parser.add_argument('-username', help='Username if credentials are required,
-passwd is also required')
parser.add_argument('-passwd', help='Password if credentials are
required',default='admin')
parser.add_argument('-z', help='Zoom STEP level 0 - 7; Default
3',default='3')
parser.add_argument('-zs', help='Do Zoom STEP defined using -z, else normal
zoom',action='store_true')
args = parser.parse_args()
print(args)
if args.ip:
  server_url='http://' args.ip
```

```
elif args.ip_address:
  server_url='http://' args.ip_address
# Set Zoom STEP
zooms=args.z;
zoomstep=args.zs;

if args.username:
  auth=HTTPDigestAuth(args.username,args.passwd)
control_cgi_url = server_url   '/cgi-bin/control.cgi'
channel = 1
headers = {'Content-Type':'application/x-www-form-urlencoded;charset=UTF-8'}
if zoomstep == True:
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_wide_
var ' zooms;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(4)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_tele_
var ' zooms;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
```

```
      )
    )
    time.sleep(4)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
else:
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_tele_
std';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(5)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
```

```
       datetime.datetime.now(),
       rs.text
      )
    )

payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_wide_
std';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
    time.sleep(5)
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_stop'
;
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
payload='action=CameraControl&interface=Visca&cam_index=1&command=zoom_get_p
os';
    rs = http_request(payload)
    print_immediately(
      "{:>6}: {}   Response: {}".format(
       1,
       datetime.datetime.now(),
       rs.text
      )
    )
```

**Command to execute with out zoom step size**

```
python3 Zoom_visca.py -ip 192.168.1.105 -username admin -passwd admin
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin', z='3', zs=False)
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":3040,"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":0,"ret":"0","status":"OK"}
```

**Command to execute with zoom step size**

```
python3 Zoom_visca.py -ip 192.168.1.105 -username admin -passwd admin -z 3 -zs
```

**Example output:**

```
(ip='192.168.1.105', passwd='admin', username='admin', z='3', zs=True)
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":0,"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"ret":"0","status":"OK"}
Response: {"response":7304,"ret":"0","status":"OK"}
```

## InsQuery

---

Querry the PTZ values for the angle.

**Parameters:**

---

***data*** Read the json table and retrieve the index and querry type. Possible querry are YawAngle, PitchAngle, RollAngle, All.

***YawAngle*** provide the yaw angle value from the PTZ.

***PitchAngle*** provide the pitch angle value from the PTZ.

***RollAngle*** provide the roll angle value from the PTZ.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## LoadUser

Load the active preset value from the Database and update in the state table database.

**Parameters:**

***enc_current_preset*** Get the preset row and check the opermation mode. The operation modes are encoder / decoder.

***cur_preset*** Provide the current presetting values.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## Login

Set the password for the login user.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## Logout

---

Logout from the system and update in the Database.

**Parameters:**

---

***Authorization*** Get the permission level.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## Overlay

---

Add/Update the overlay values in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***rgn_idx*** Index to map overlay. The region index value can be in the range of 1 to 16. Ambarella provides 16 total color look-up tables for Overlay, rgn_idx signifies which color look-up table is in use.

***type*** Type of overlay to be used possible values: text or png.

***source*** In the case of text overlay this would be the source text for png overlay this is the path to the png on the encoder. Images must be uploaded to board to be used.

***location*** This is the location of the overlay. Possible values: 'top_left', 'top_right', 'top_center', 'bottom_left', 'bottom_right', 'bottom_center', 'x,y' (negative numbers not supported for x or y).

***char_size*** For text overlay this is the character size. Possible values: 16,32,64, 128.

***layer*** This will set the layer for the overly higher numbers will overlay over lower numbers if overlapping.

***alpha*** Sets the transparency of the text. Possible values: 0-255.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## OverlayStop

---

Stop overlay.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***rgn_idx*** Index of overly for this channel.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzAbsoluteMove

---

Send absolute move command to the PTZ.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzAuxiliary

Send PTZ auxiliary value to the PTZ.

**Parameters:**

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## PtzClearPreset

Remove the Ptz preset values for the user.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position and speed value. The mode values can be Pan, tilt, and Zoom.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzContinuousMove

---

Send continous move command to the PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzGotoPreset

---

Send preset value to PTZ.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzNewTourSpot

---

Get the new tour spot and update in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzPosition

---

Move the Ptz to the input position

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzPreset

---

Preset the Ptz based on the value.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzQuery

---

Get the ptz status and degree value from the ptz

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzRemoveTourSpot

---

Remove the ptz tour spot

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzSetPreset

---

Set the preseting value in the PTZ

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzStartTour

---

Start the PTZ tour

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## PtzStop

---

Stop the PTZ

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the mode, position, speed value, ptz_timeout, profile token.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## RemoveJobs

---

Delete the schedule jobs from the Database.

**Parameters:**

---

***purgelist*** List of schedule jobs to be removed.

***rowcnt*** Number of rows.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## RestartBoard

---

Read the operation mode. Whether its encoding / decoding. Stop the channel encoding and restart

the board

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveCamera

---

Saving the camera settings in the Database.

**Parameters:**

---

***zoom_direct_value*** Zoom value. Possible values: 0 (wide) to 0x7ac0 (full zoom)

***white_balance_mode*** White balance mode. Default value is auto. Possible values: auto / manual / indoor/ one-push / auto trace / sodium lamp auto / sodium lamp fixed / sodium lamp outdoor.

***color_gain*** color gain. Default value is 4. Possible values: Range from 0 (60%) to 14 (200%)

***color_hue*** color hue. Default value is 7. Possible values: Range from 0 (60%) to 14 (200%)

***chroma_suppress*** Chroma suppress. Default value is 1. Possible values: 0=none, 1, 2, 3

***wb_manual_rgain_direct*** white balance manual red gain. Default value is 190. Possible values: 0 to 255

***wb_manual_bgain_direct*** white balance manual blue gain. Default value is 190. Possible values: 0 to 255

***optical_zoom_only*** optical zoom only. Default value is 0. Possible values: 0, 1

***focus_direct_value*** focus direct value. Default value is 4096. Possible values: 0x1000 to 0xf000

***manual_focus*** Manual focus. Default value is auto. Possible values: auto, manual

***exposure_mode*** Exposure mode. Default value is 0. Possible values: 0 - Auto, 10 - Shutter Priority,

11 - Iris -Priority, 3 - Manual.

***shutter*** Shutter. Default value is 7.

***iris*** Iris. Default value is 0.

***gain*** Gain. Default value is 0.

***high_sensitivity*** high sensitivity value. Default value is 3. Possible values 0, 1.

***hlc_level*** hlc level. Default value is 0. Possible values in range 0 to 3.

**hlc_level_mask** mask value of hlc level. Default value is 0. Possible values 0, 1.

***stable_zoom*** stable zoom on or off. Default value is off.

***eflip*** Default value is 5. Possible values 0, 1.

***lr_reverse*** Default value is 0. Possible values 0, 1.

***monitor_mode*** Default value is 1080p-59.94.

***genlock_source*** Default value is 2.

***zoom_step_size*** Default value is 4. Possible values in range (1 - 7).

***focus_step_size*** Default value is 1. Possible values in range (0 - 7).

***manual_icr*** Default value is Off. Possible values: On, Off, Auto, Auto Color

***img_freeze*** Default value is 0. Possible values: 0, 1.

***hr_mode*** High resolution mode. Default value is 0. Possible values: 0, 1.

***img_stabilizer*** Default value is 0. Possible values 0, 1.

***img_bw*** Image black white. Default value is 0. Possible values 0, 1.

***nr_2d_level*** Default value is 5. Possible values in range (0 - 5).

***nr_3d_level*** Default value is 4. Possible values in range (0 - 5).

***icr_threshold*** Default value is 14. Possible values in range (0-255 for 4K camera, 0-28 for HD camera).

***slow_shutter*** Default value is 0. Possible values: 0, 1.

***slow_shutter_limit*** Default value is 4. Possible values in range: (1 to 6)

***flicker_reduction*** Default value is 0. Possible values: 0, 1.

***img_stabilizer_level*** Default value is 0. Possible values: 0, 2.

***wide_dynamic_range*** Default value is 3.

***ve_brightness*** Default value is 3. Possible values in range: 0, 6.

***ve_compensation_type*** Default value is 2. Possible values in range: 0, 3.

***ve_compensation_level*** Default value is 1. Possible values in range: 0, 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveCameraBoson

---

Save color pallete settings of boson camera in the Database.

**Parameters:**

---

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveCameraDRSTamarisk

---

Save color pallete settings of DRS Tamarisk camera in the Database.

**Parameters:**

---

***color_pallette*** Color pallete. Default value is whitehot.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveCameraLink

---

Saving camera link settings in the Database.

**Parameters:**

---

***pxl_format*** Pixel format. Default value is 1.

***color_table*** Color table. Default value is whitehot.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveCameraLx

---

Saving camera Lx settings in the Database.

**Parameters:**

---

***SONY_ExposureMode*** Exposure Mode. Default value is 32852. Possible values: 32848 (Program Auto), 32849 (Aperture Priority), 32850 (Shutter Priority), 32851 (Manual Exposure), 32852 (Intelligent Auto).

***SONY_ShutterSpeed*** Shutter speed. Defualt value is NULL.

***SONY_ISO*** ISO. Defualt value is NULL.

***SONY_ExposureComp*** Exposure comp. Defualt value is NULL.

***SONY_WhiteBalance*** White balance. Defualt value is NULL.

***SONY_ColorTemp*** Color temperature. Defualt value is NULL.

***SONY_APS_C*** APS_C. Defualt value is NULL.

***SONY_NtscPalSelect*** NTSC / PAL selection. Defualt value is NULL.

***SONY_MovieSteadyMode*** Movie steady mode. Defualt value is NULL.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveCameraTau

---

Saving camera TAU settings in the Database.

**Parameters:**

---

***color_table*** color table value .

***is_active*** Active/Deactive. Default value is 0.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveJobs

---

Save the schedule jobs in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the schedule activities.***.***

***rowcnt*** Get the number of row count.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SavePresets

---

Check the operation mode whether its in encoder or decoder. Save the preseting value of the mode in the Database.

**Parameters:**

---

***dec_current_preset*** Save the decoder current preseting values.

***enc_current_preset*** Save the encoder current preseting values.

***preset*** Settings for encoder or decoder.

***enc_channels*** Number of channels. Maximum number of channels is 2 for decoder and 4 for encoder.

***opmode*** Operation mode. Default is Encoder. Possible values: Encoder / Decoder.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SaveUser

---

Save the state, preset, encoder/decoder settings in the Database.

**Parameters:**

---

***enc_current_preset*** Get the encoder preseting values for the current channel.

***dec_current_preset*** Get the decoder preseting values for the current channel.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetAdv

---

Save the encoder advance settings in the Database.

**Parameters:**

---

***enc_adv_setting*** Get the encoder advance setting values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetAPConfig

---

Set the wifi and its password.

**Parameters:**

---

***passwd*** Get the wifi password

***passwden*** Get wifi enable / disable. Default value is disable.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetAudio

Save the audio input settings in the Database.

**Parameters:**

***aport*** Get the audio port. Default value is MIC. Possible values are MIC , MICL

***analog_gain_db*** Get the analog gain decibal value. Default value is 0. The value ranges are -97 to 30.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetCameraLink

Save the color table and pixel format values in the Database and set the color table and pixel format in the fpga.

**Parameters:**

***val*** Get the value for color table and pixel format.

***opt*** The options are color table and pixel format.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetConsole

Set the console value as ttyAMA0 or none.

**Parameters:**

***console_enable*** Get console value is enabled / disabled.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetDDNSEnable

Set the Dynamic Domain Name System settings in the Database.

**Parameters:**

***ddns_enable*** Get the ddns enable value. Default value is Off. Possible values; (On, Off)

***ddns_provider*** Get the ddns provider. Default value is freedns. Possible values: *(*freedns, freemyip, dyn, domains.google.com, duckdns.org, no-ip.com, tunnelbroker.net, dynv6.com, cloudxnv.net, dnspod.cn, cloudflare.com).

***ddns_username*** Provide the username.

***ddns_password*** Provide the password.

***ddns_hostname*** Hostname for DDNS login.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetDebugLevel

---

Set the logging level. The setting is persistent.

**Parameters:**

---

***sysdebuglevel*** Get the system debug level. Possible values 0=None, 1=Error, 2=Information, 3=Diagnostics, 4=Codeflow, 5=Dataflow.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetDevName

---

Set the device name in the Database.

**Parameters:**

---

***sysdevicename*** Get the system device name.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetDisplay

---

Controls composite output (passthru video from camera).

**Parameters:**

---

***disp_std*** Get display standard for encoder. Default value is Auto.

***disp_input*** Get display input for encoder. Default value is MACRO_DEFAULT_DISP_INPUT.

***save_only*** Possible values: (True / False)

***disp_layout*** Display layout for decoder. Default value is 1×1.

***disp_mode*** Display mode for decoder. Default value is UltraHD.

***disp_std2*** Display standard for decoder. Default value is auto.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetDSCP

---

Set the DSCP configuration settings in the Database.

**Parameters:**

---

***diff_serve*** Get the DSCP configuration value. Default value is 0x00. Possible values are ( 0x00, 0xB8, 0x88, 0x90, 0x98, 0x80, 0x68, 0x70, 0x78, 0x60, 0x48, 0x50, 0x58, 0x28, 0x30, 0x38, 0x01).

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetFpga

---

Set the fpga settings in the Database.

**Parameters:**

---

***fpgafileglob*** Get fpga value. Possible values are (boson / cvbs / tamarisk / tau2).

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetIp

---

Control networking settings of the Z3 device such as local IPv4 IP address, netmask, etc. Additionally control Encoder Auto-Start after boot-up and enable/disable showing of advanced WebUI settings.

**Parameters:**

---

***do_autostart*** Control automatic stream start after bootup. Possible values: 1 = do autostart 0 = do not autostart.

***enc_adv_setting*** Control appearance of advanced settings on WebUI. Possible values: off, on.

***ipmtu*** Control Ethernet IP Maximum Transmission Unit (MTU) size in bytes. Possible values: 576 to 1500.

***eth_speed*** Control Gigabit Ethernet auto-negotiation allowed speeds. Possible values: AUTO, AUTO-100, 1000, 100, 10.

***eth_duplex*** Control Ethernet auto-negotiation allowed duplex. Possible values: AUTO, FULL, HALF.

***local_ip*** IPv4 address.

***local_netmask*** IPv4 netmask.

***default_gw*** IPv4 default gateway.

***local_dnsip*** DNS server primary.

***local_dnsip2*** DNS server secondary.

***use_dhcp*** Controls use of DHCP or static IP. Possible values: 1 (DHCP enabled), 0 (DHCP disabled, use static IP).

***local_hostname*** Network hostname.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetLoginLimit

---

Set the login limit settings in the Database

**Parameters:**

---

***maxlogin*** mximum number of login limit. Possible value in range (3 to 99 ).

***login_window*** Login window period, the value are considered in seconds. Possible value in range (60 to 32768).

***lockout_interval*** Lockout interval values are mentioned in seconds, which is used when maximum number of login limit is reached. Possible value in range (-1 to 32768)

***login_timeout*** Login timeout

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetNFS

---

Set the NFS settings in the Database.

**Parameters:**

---

***nfs_enable*** Enable/Disable nfs

***nfs_server*** Get the nfs server ip address. Default is 192.168.1.6

***nfs_server_root*** Get the server root path / location.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetSNTP

Set SNTP (Simple Network Time Protocol) settings in the Database.

**Parameters:**

***enable*** Enable/Disable SNTP. Possible value: true, false. Default value true.

***servers*** NTP server or list of NTP servers.

***timezone*** Linux TZ database value for Timezone.

***timezone_name*** Name for a time zone. Default is America/Chicago.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetSNMP

Set the SNMP settings in the database.

**Parameters:**

***power_trap_en*** Enable/Disable power trap.

***input_loss_trap_en*** Enable/Disable input loss trap.

***input_recover_trap_en*** Enable/Disable input recover trap.

***temp_high_trap_en*** Enable/Disable temp high trap.

***temp_recover_trap_en*** Enable/Disable temperature recover trap.

***trap_hosts*** Trap hosts ipaddress. Default ip address value is 192.168.0.6.

***high_temp_suppress*** hiigh temperature. Default value is 30.

***nominal_temp_suppress*** nominal temperature default value is 30.

***snmp_v3_en*** Enable/Disable SNMP

***snmp_v3_encrypt*** SNMP v3 encrypt value.

***snmp_v3_user*** z3user.

***snmp_v3_passwd*** z3password.

***snmp_v3_usrpro*** snmp v3 usrpro default value is md5.

***snmp_v3_encrypt_passwd*** z3password.

***snmp_v3_encpro*** decrypt value.

***snmp_v3_engineid*** Default value for engineid is 0.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetNMEAEnable

---

Save the GPS NMEA enable/disable value in the Database.

**Parameters:**

---

***nmea_enable*** Enable/Disable the NMEA. Possible value: on, off.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetOnvif

Save the ONVIF settings in the Database.

**Parameters:**

***fixed_profile_max*** Get the Maximum ONVIF profiles to allow (1 or 2).

***en_persistent_audio*** Enable/Disable ONVIF persistent audio channel.

***onvif_enable*** Enable/Disable ONVIF.

***list_all_video_sources*** List all the video source.

***ptz_timeout*** Get the PTZ timeout values.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetOnvifVMD

Save the ONVIF video motion detection and too darkness settings in the Database.

**Parameters:**

***vmd_enable*** Enable/Disable video motion detection.

***vmd_sens*** sensitivity value required for vmd. Possible value in range: (1 to 100).

***vmd_zone_modify*** Enable/Disable video motion and too darkness zone coordinates.

***vmd_vcrop_width*** crop width of the coordinates.

***vmd_vcrop_height*** crop height for VMD and too_darkness.

***vmd_vcrop_x*** crop startx for VMD and too_darkness.

***vmd_vcrop_y*** crop starty for VMD and too_darkness.

***td_enable*** Enable/Disable too darkness.

***td_thresh*** Threshold value for too darkness. Possible values in range: 1 to 100.

***vmd_td_channel*** Get the channel number. Possible values: 1, 2.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetRTSP

---

Set the RTSP settings in the Database.

**Parameters:**

---

***rtspd_port*** Get rtsp port. Default value is 554.

***rtspd_timeout*** Timeout value required for the rtsp connection.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetTermSrvEnable

Sett the term server value in the Database.

**Parameters:**

***termserve_remote_enable*** Enable/Disable remote access to terminal server. Possible values are : on, off.

***termserve_shared_enable*** Enable/Disable shared access to terminal server. Possible values are : on, off.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetVideoGroup

Set the video group configuration settings in the Database.

**Parameters:**

***group*** Video config groups. Default value 0.

***nr_enable*** Enable / Disable NR. Default value is on. Possible value: on/off.

***crop_enable*** Enabled / Disable the crop.

***crop_x*** Crop x coordinates.

***crop_y*** Crop y coordinates.

***crop_width*** Crop width

***crop_height*** Crop height

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetViewport

---

Set the view port for the decoder and update in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***viewport*** viewport settings. The default value is fit.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetViVpssMode

Set the ViVpss mode settings in the Database.

**Parameters:**

***enc_vivpss_mode_enable*** Enable/Disable the vivpss mode. Default value is off. Possible values: on, off.

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## SetWifiAP

Set the wifi network settings in the Database.

**Parameters:**

***ssid*** Get the Wifi service set identifier value.

***psk*** Get the Wifi password.

***wifi_client_local_ip*** Local IP addres.

***wifi_client_local_local_netmask*** Subnet mask for Wifi network.

***wifi_client_default_gw*** Default Gateway be used for Wifi Network.

***wifi_client_use_dhcp*** DHCP server used with Wifi network for DNS resolutions.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## SetZFinderEnable

---

Save ZFinder enable / disable in the Databse.

**Parameters:**

---

***zfinder_enable*** Get ZFinder enable/disable value. Possible values: off, on

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StartChannel

---

Start the encoder channel and update the state as running in the Database. Once you start the channel, it will not transmit data until the video input is detected.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

***actv_preset***

***vsource*** video source. Default value is Camera1. Possible values: Camera1 / Camer2 / HDMI1.

***vres*** video resolution. Default value is Follow Input. Possible values: Follow Input, 3840 x 2160, 1920 x 1080, 1440 x 1080, 1280 x 1024, 1280 x 720, 1024 x 768, 960 x 720, 1024 x 576.

***vcodec*** video codec. Default value is H265. Possible values: H265, H264, MJPEG, none.

***vgdr*** video gradual data refresh. Default value is on. Possible value on, off.

***vprofile*** video profile. Default value is high. Possible value high, main, baseline.

***vratectrl*** video rate control. Default value is cbr. Possible value cbr, vbr.

**vbitrate** video bit rate.

***vframeratediv*** video frame rate. Default value is 1. Possible values: 1 (Full), 2 (Half), 4 (Quarter), 6 (Sixth).

***vgopsize*** video group of picture size. Default value is 60. Possible values: 1 (1 Frame Only), 5 (5 Frames), 8 (8 Frames), 10 (10 Frames), 12 (12 Frames), 15 (15 Frames), 25 (25 Frames), 30 (30 Frames), 50 (50 Frames), 60 (60 Frames), 90 (90 Frames), 100 (100 Frames), 120 (120 Frames), 200 (200 Frames), 240 (240 Frames).

***vprotocol*** video protocol.

***vdest*** video destination port.

***vpid*** video packet identifier. Default value is 221.

***vdelay*** video delay. Default value is 300.

***pcrpid*** PCR packet identifier. Default value is 521.

***pcrinterval*** Default value is 50.

***pmtpid*** program map table packet identifier. Default value is 31.

***tsrate*** bit rate for transport stream. Default value is 5000K.

***tslowlat*** transport stream low latency mode. Default value is off. Possible values: on, off.

***feconoff*** Forward error correction in transport stream enable/ disable. Default value is off. Possible values: on, off.

***fecrow*** Forward error correction row. Default value is 1.

***feccol*** Forward error correction column, Default value is 5.

***zixioverhead*** ZIXI Forward Error Correction total overhead as a percentage of transport rate. Default value is 15.

***zixilatency*** ZIXI target latency (to allow ARQ) in milliseconds. Default value is 500.

***zixifecblock*** ZIXI Forward Error Correction block size in milliseconds. Max value is 1/2 of Zixi Latency. Default value is 50.

***zixirateadjen*** ZIXI dynamic bitrate adjustment according to network conditions. Default value is on. Possible values: on, off.

***zixiauthen*** ZIXI dynamic authentication enable/disable . Default value is off. Possible values: on, off.

***zixisession*** ZIXI session id. Default value is test.

***zixiuser*** ZIXI user identifier. Default value is user.

***aenable*** Audio enable/disable. Default value is on. Possible values: on, off.asource. Default value is MICL.

***apair*** Stereo pair select for digital inputs. Default value is 0. Possible values: (0 (1+2), 1 (3+4), 2 (5+6), 3 (7+8), 260 (1+2,3+4), 548 (1+2,3+4,5+6), 1252 (1+2,3+4,5+6,7+8)).

***acodec*** Audio codec. Default value is fdk_aaclc.

***abitrate*** Audio bit rate. Default value is 128000.

***asamplerate*** Audio sample rate. Default value is 48000.

***aport*** Audio port. Default value is 8700.

***apid*** Audio packet identifier. Default value is 120.

***aptspcr*** Maximum difference between the PTS and PCR in the audio stream Default value is 250.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StartMTS

---

Start the MTS.

**Parameters:**

---

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StopChannel

---

Stop the encoder channel and update the state in the Database.

**Parameters:**

---

***chn*** Get the encoder channel number. The default is encoder channel 1. Select between encoder channels. Possible values: 1, 2, 3, or 4.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## StopMTS

---

Stop the MTS.

**Parameters:**

---

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## TempStatus

Get the current temperature status for CPU and lense.

**Parameters:**

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

## update_require_web_login

Update/Change whether web login credential required or not in the Database.

**Parameters:**

***require*** Required web login value. Possible values: 1 or 0

**Returns:**

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## UpdatePtz

---

Update the PTZ value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## UpdatePtzPreset

---

Update the PTZ preset value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ preset values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---

## UpdatePtzTourSpots

---

Update the PTZ tour spots value in the Database.

**Parameters:**

---

***data*** Parse the JSON formatted data and retrieve the PTZ tour spots values.

**Returns:**

---

JSON formatted table with values below

***ret*** Custom return code of the HTTP POST request. 0 on success, 1 on failure.

***status*** String describing the failure when **ret** is 1. “OK” on success.

---