Contents

1 Overview 5
1.1 Design Points 5
1.2 Connection Scenario 5
1.3 Status Code 5

2 Authentication 6
2.1 Request 6
2.2 Response 6
2.3 Sample 6

3 API Functions 6
3.1 Amplifier Name 6
3.1.1 Request 6
3.1.2 Response 7
3.2 IP Settings 7
3.2.1 Request 7
3.2.2 Response 7
3.3 Input Name 8
3.3.1 Request 8
3.3.2 Response 9
3.4 Input Volume(Input Level Trim) 9
3.4.1 Request 9
3.4.2 Response 10
3.5 Output Name 10

3.5.1 Request 10
3.5.2 Response 11
3.6 Output Selection(Source1) 11
3.6.1 Request 11
3.6.2 Response 11
3.7 Output Selection(Source2) 12
3.7.1 Request 12
3.7.2 Response 12
3.8 Bridge Mode 12
3.8.1 Request 12
3.8.2 Response 13
3.9 Stereo/Mono Output 13
3.9.1 Request 13
3.9.2 Response 14
3.10 MuteOutput Output 14
3.10.1 Request 14
3.10.2 Response 14
3.11 Output Volume 15
3.11.1 Request 15
3.11.2 Response 15
3.12 Turn On Volume Per Output 16
3.12.1 Request 16
3.12.2 Response 16
3.13 Max Volume Per Output 16
3.13.1 Request 16
3.13.2 Response 17
3.14 DSP Presets Per Output 17
3.14.1 Request 17
3.14.2 Response 18
3.15 Output Group 18
3.15.1 Request 18
3.15.2 Response 18
3.16 Get EQ Settings 19

3.16.1 Request 19
3.16.2 Response 19
3.17 Set EQ Settings 20
3.17.1 Request 20
3.17.2 Response 20
3.18 Add EQ Settings 20
3.18.1 Request 20
3.18.2 Response 21
3.19 Delete EQ Settings 21
3.19.1 Requst 21
3.19.2 Response 21
3.20 Low Tilt Per Output 21
3.20.1 Request 21
3.20.2 Response 22
3.21 High Tilt Per Output 23
3.21.1 Request 23
3.21.2 Response 24

4 Local Configuration Tool Only 25
4.1 Factory Reset 25
4.1.1 Request 25
4.1.2 Response 25
4.2 Reboot 25
4.2.1 Request 25
4.2.2 Response 26
4.3 Auto-On Method Abandon 26
4.3.1 Request 26
4.3.2 Response 26
4.4 Audio Sense Method 26
4.4.1 Request 26
4.4.2 Response 27
4.5 Power Button Method 27
4.5.1 Request 27
4.5.2 Response 27

4.6 Sleep Mode 28
4.6.1 Request 28
4.6.2 Response 28
4.7 Mode Source2 29
4.7.1 Request 29
4.7.2 Response 29
4.8 Gain Offset 29
4.8.1 Request 29
4.8.2 Response 30
4.9 Test Signal 30
4.9.1 Request 30
4.9.2 Response 30
4.10 Low Pass 31
4.10.1 Requeset 31
4.10.2 Response 32
4.11 High Pass 33
4.11.1 Request 33
4.11.2 Response 34
4.12 Delay 35
4.12.1 Request 35
4.12.2 Response 36
4.13 Limiter 37
4.13.1 Request 37
4.13.2 Response 37
4.14 Copy DSP Preset 37
4.14.1 Request 37
4.14.2 Response 38

5 Unsolicited Message 38

Appendices 38

Appendix A Access Local WEB UI HOWTO 38

Appendix B Cheat Sheet 39

B.1 Avoid Port Confliction 39
B.2 http://192.168.1.200 instead of http://127.0.0.1 39

1 Overview

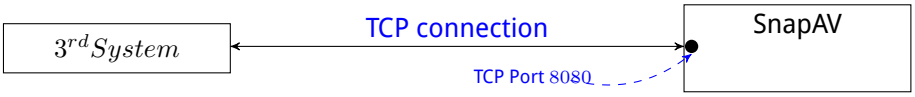
1.1 Design Points

- TCP connection is used for $3^{rd} System$ controlling target device.
- Auth required for each TCP connection.
- Control data is in C/C++ style string, i.e., NULL-terminated string.
- String is encoded with UTF-8, formatted in JSON.

1.2 Connection Scenario

TCP server is listening at SnapAV Amplifier side.

$3^{rd} System$ is a TCP client:



Note:
Port 8080 can be config-ed dynamically

The TCP connection would support up to 10 multiple simultaneous connections.

1.3 Status Code

All data from Amplifier device can be indicated via status field of the JSON, which is an integer value.

Below is current definition:

Code Value	Meaning	Code Value	Meaning
200	Success	300	Repeat operation
400	User or password error	401	Illegal request
402	please log in first	403	The password is default and needs to be changed
405	Account is signed out	406	Too many incorrect passwords, account locked
407	Enter the same password as before, please reset	408	Password set successfully
410	Invalid data range		
500	An unknown error occurred on the server	600	The import file does not exist
601	File MD5 values do not match	602	File type mismatch
700	The machine is in standby mode	701	The machine is in Voltage Trigger mode
702	The machine is in Audio mode	800	Update success
801	Update fail	900	discarded protocol

2 Authentication

2.1 Request

The auth account is the same as WEB UI's.

The format of request:

```
{"type":"login","username":"name","password":"passwd"}
```

2.2 Response

```
{"type":"login","status":200}
```

The status indicates the result of response.

2.3 Sample

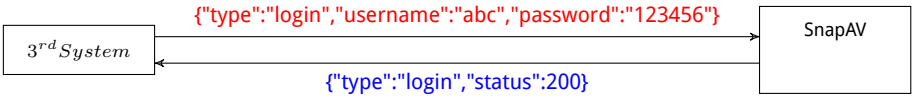
Suppose we have below user account info:

Username	Password
abc	123456

Then, the auth process is like this:

Example 1

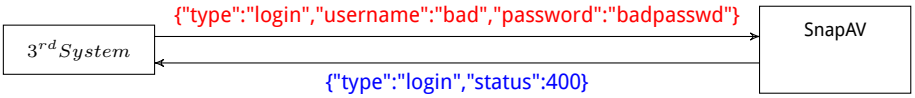
Authentication of the TCP connection



For incorrect auth case:

Example 2

Faiure of Authentication



3 API Functions

3.1 Amplifier Name

3.1.1 Request

Change the amplifier name:

```
{"type":"set_ampname","value":"new amplifier name"}
```

Get the amplifier name:

```
{"type":"get_ampname"}
```

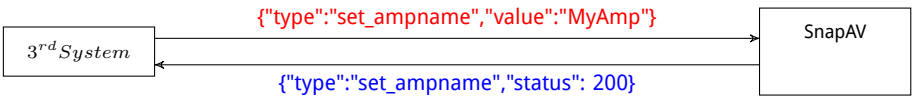
3.1.2 Response

For set case:

```
{"type":"set_ampname","status": 200}
```

Example 3

Set Amplifier Name to MyAmp



For get cases:

```
{
    "type":"get_ampname",
    "status": 200,
    "value":"your amplifier name"
}
```

3.2 IP Settings

3.2.1 Request

Change the IP setting:

```
{
  "type":"set_ip",
  "value": {
    "dhcp": "on/off",
    "ipaddress":"192.168.199.100",
    "gateway":"192.168.199.1",
    "subnetmask":"255.255.255.0",
    "dnsserver":"192.168.199.1"
  }
}
```

Get the IP setting:

```
{"type":"get_ip"}
```

3.2.2 Response

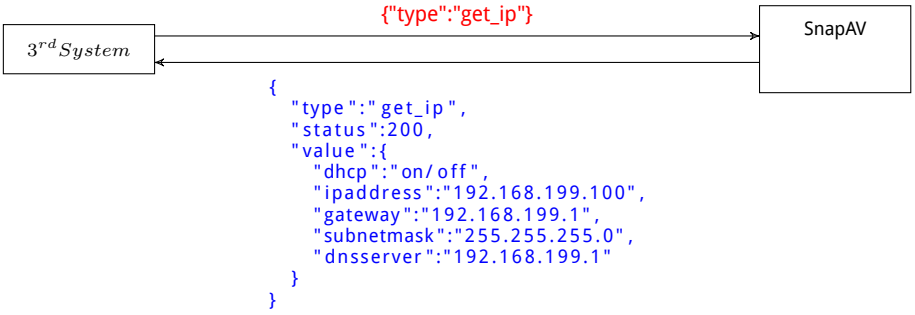
For set cases:

```
{"type":"set_ip","status":200}
```

For get case, check below sample:

Example 4

Get IP Settings Sample Output



Check Section-1.3(Page-5) for more details of status value meaning.

3.3 Input Name

3.3.1 Request

Set:

```
{
  "type":"set_inputname",
  "value":[
    {"index": id, "name": "input name value"}
  ]
}
```

Get:

```
{
  "type":"get_inputname",
  "value":[
    id1, id2, id3 ...
  ]
}
```

The index is numerical value, indicating which channel to be operated.

Suppose Amplifier has N channels, then the $index$ numerical definition follows below rule:

$$0 \leqslant index \leqslant (\frac{N}{2} - 1) \tag{1}$$

Example 5

Valid index range for different channel products

$$
\begin{aligned}
2d: &\quad 0 \leqslant index \leqslant 0 \\
8d: &\quad 0 \leqslant index \leqslant 3 \\
12d: &\quad 0 \leqslant index \leqslant 5 \\
16d: &\quad 0 \leqslant index \leqslant 7
\end{aligned}
$$

3.3.2 Response

For set cases:

```
{
  "type":"set_inputname",
  "status": 200
}
```

For get cases:

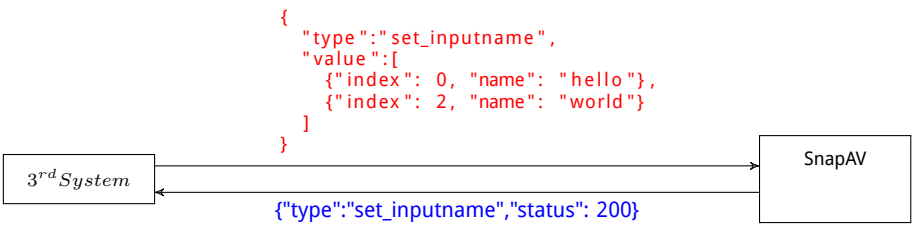
```
{
  "type":"get_inputname",
  "status":200,
  "value":[
      {"index":id1, "name":"id1 input name"},
      {"index":id2, "name":"id2 input name"}
      ... ...
  ]
}
```

Check Section-1.3(Page-5) for more details of status value meaning.

Suppose we want to set 1^{st} and 3^{rd} input channel name on 12d product:

Example 6

Set input name on 12d



3.4 Input Volume(Input Level Trim)

3.4.1 Request

For set:

```
{
  "type":"set_inputvol" ,
  "value":[
    {"index": id1 , "value": "input value1"},
    {"index": id2 , "value": "input value2"}
  ]
}
```

For get:

```
{
  "type":"get_inputvol" ,
  "value":[
    id1 , id2, ...
  ]
}
```

The index is numerical value, indicating which channel to be operated. The numerical value follows Equation-(1)

3.4.2 Response

For set cases:

```
{
  "type":"set_inputvol",
  "status": 200
}
```

For get cases:

```
{
  "type":"get_inputvol",
  "status":200,
  "value":[
      {"index":id1, "value":"id1 input vol"},
      {"index":id2, "value":"id2 input vol"}
        ... ...
  ]
}
```

3.5 Output Name

3.5.1 Request

Set:

```
{
    "type":"set_outputname" ,
    "value":[
      {"index": id1 , "value": "output name value"}
      {"index": id2 , "value": "output name value"}
    ]
}
```

Get:

```
{
  "type":"get_outputname",
  "value":[
    id1, id2, ...
  ]
}
```

3.5.2 Response

For set cases:

```
{
  "type":"set_outputname",
  "status": 200
}
```

For get cases:

```
{
  "type":"get_outputname",
  "status":200,
  "result":[
    {"index":id1, "value":"id1 output name"},
    {"index":id2, "value":"id2 output name"}
    ... ...
  ]
}
```

3.6 Output Selection(Source1)

3.6.1 Request

Set:

```
{
  "type": "set_outputsource1" ,
  "value":[
    {"index": id , "value": "input value index "}
  ]
}
```

Get:

```
{
  "type": "get_outputsource1" ,
  "value":[ id1, id2, ...]
}
```

3.6.2 Response

For set case:

```
{
  "type": "set_outputsource1" ,
  "status":200
}
```

For get case:

```
{
  "type": "get_outputsource1" ,
  "status":200,
  "value":[
    {"index":id1, "value":"id1 source1 input index"},
    {"index":id2, "value":"id2 source1 input index"},
    ... ...
    {"index":idx, "value":"idx source1 input index"}
```

```
  ]
}
```

3.7 Output Selection(Source2)

3.7.1 Request

Set:

```
{
  "type": "set_outputsource2" ,
  "value":[
    {"index": id , "value": "input value index "}
  ]
}
```

Get:

```
{
  "type": "get_outputsource2" ,
  "value":[ id1, id2, ... ]
}
```

3.7.2 Response

For set:

```
{
  "type": "set_outputsource2" ,
  "status":200
}
```

For get:

```
{
  "type": "get_outputsource2" ,
  "status":200,
  "value":[
        {"index":id1, "value":"id1 source2 input index"},
        {"index":id2, "value":"id2 source2 input index"},
        ... ...
        {"index":idx, "value":"idx source2 input index"}
  ]
}
```

3.8 Bridge Mode

3.8.1 Request

set:

```
{
  "type":"set_bridgemode" ,
  "value": [
     {"index": id , "value": "0 or 1"}
  ]
}
```

get:

```
{
  "type":"get_bridgemode",
  "value":[id1, id2, ...]
}
```

The index is numerical value, indicating which channel to be operated. The numerical value follows Equation-(1)

The bridge value definition:

value	Meaning
"0"	mode off
"1"	mode on

3.8.2 Response

For set:

```
{
  "type":"set_bridgemode",
  "status": 200
}
```

For get:

```
{
    "type":"get_bridgemode",
    "status":200,
    "value":[
        {"index":id1, "value":"0 or 1"},
        {"index":id2, "value":"0 or 1"},
        ... ...
    ]
}
```

3.9 Stereo/Mono Output

3.9.1 Request

Set:

```
{
  "type":"set_stereomono" ,
  "value":[
    {"index": id , "value": "0 or 1"}
  ]
}
```

Get:

```
{
  "type":"get_stereomono" ,
  "value":[ id1, id2, ..., idx ]
}
```

The stereo/mono value defintion:

value	Meaning
"0"	stero
"1"	mono

3.9.2 Response

For set:

```
{
  "type":"set_stereomono",
  "status": 200
}
```

For get:

```
{
  "type":"get_stereomono",
  "status": 200,
  "value":[
        {"index":id1, "value":"0 or 1"},
        {"index":id2, "value":"0 or 1"},
        ... ...
        {"index":idx, "value":"0 or 1"}
  ]
}
```

3.10 MuteOutput Output

3.10.1 Request

Set:

```
{
  "type":"set_muteoutput" ,
  "value":[
    {"index": id , "value": "0 or 1"}
  ]
}
```

Get:

```
{
  "type":"get_muteoutput" ,
  "value":[ id1, id2, ... ]
}
```

The value definition:

value	Meaning
"0"	off
"1"	on

3.10.2 Response

For set:

```
{
  "type":"set_muteoutput",
  "status": 200
}
```

For get:

```
{
  "type":"get_muteoutput",
  "status": 200,
  "value":[
        {"index":id1, "value":"0 or 1"},
        {"index":id2, "value":"0 or 1"},
        ... ...
        {"index":idx, "value":"0 or 1"}
  ]
}
```

3.11 Output Volume

3.11.1 Request

set:

```
{
  "type":"set_outputvol" ,
  "value":[
    {"index": id1 , "value": "output value1"},
    {"index": id2 , "value": "output value2"}
  ]
}
```

Get:

```
{
  "type":"get_outputvol" ,
  "value":[ id1, id2, ... ]
}
```

The index is numerical value, indicating which channel to be operated. The numerical value follows Equation-(1)

3.11.2 Response

For set:

```
{
  "type":"set_outputvol",
  "status": 200
}
```

For get:

```
{
  "type":"get_outputvol",
  "status": 200,
  "value": [
    {"index": id1, "value": "val"},
    {"index": id2, "value": "val"},
    ... ...
    {"index": idx, "value": "val"}
  ]
}
```

3.12 Turn On Volume Per Output

3.12.1 Request

Set:

```
{
  "type":"set_turnonvol" ,
  "value":[
     {"index": id , "value": "turn on volume value"}
  ]
}
```

Get:

```
{
  "type":"get_turnonvol",
  "value":[id1, id2, ..., idx]
}
```

3.12.2 Response

For set:

```
{
  "type":"set_turnonvol" ,
  "status": 200
}
```

For get:

```
{
    "type":"get_turnonvol",
    "status": 200,
    "value":[
        {"index":id1, "value":"id1 turn on volume"},
        {"index":id2, "value":"id2 turn on volume"},
        ... ...
        {"index":idx, "value":"idx turn on volume"}
    ]
}
```

3.13 Max Volume Per Output

3.13.1 Request

Set:

```
{
  "type":"set_maxvol" ,
  "value": [
     {"index": id , "value": "Max vol value"}
  ]
}
```

Get:

```
{
    "type":"get_maxvol",
    "value":[id1, id2, ..., idx]
}
```

3.13.2 Response

For set:

```
{
  "type":"set_maxvol",
  "status": 200
}
```

For get:

```
{
    "type":"get_maxvol",
    "value":[
        {"index":id1, "value":"id1 max volume"},
        {"index":id2, "value":"id2 max volume"},
        ... ...
        {"index":idx, "value":"idx max volume"}
    ]
}
```

3.14 DSP Presets Per Output

3.14.1 Request

Set:

```
{
  "type":"set_dsppreset",
  "value ":[
    {"index": id , "value": "dsp preset item"}
  ]
}
```

get:

```
{
    "type":"get_dsppreset",
    "value":[id1, id2, ..., idx]
}
```

The value indicates DSP preset item, defintion:

value	Meaning
"0"	FLAT
"1"	CLASSIC
"2"	POP

Table 1: DSP Preset Item Definition

3.14.2 Response

For set:

```
{
  "type":"set_dsppreset",
  "status": 200
}
```

For get:

```
{
    "type":"get_dsppreset",
    "value":[
        {"index":id1, "value":"0 or 1 or 2"},
        {"index":id2, "value":"0 or 1 or 2"},
        ... ...
        {"index":idx, "value":"0 or 1 or 2"}
    ]
}
```

3.15 Output Group

3.15.1 Request

Set:

```
{
  "type":"set_outputgroup" ,
  "value":[
    {"index": id , "value": "group value"}
  ]
}
```

Get:

```
{
    "type":"get_outputgroup",
    "value":[id1, id2, ..., idx]
}
```

The index is numerical value, indicating which channel to be operated. The numerical value follows Equation-(1)

The groups are defined as $A, B, C, D, E, \cdots$, and the field value defintion:

value	Meaning	value	Meaning
"0"	Group-A	"1"	Group-B
"2"	Group-C	"3"	Group-D

3.15.2 Response

For set:

```
{
  "type":"set_outputgroup",
  "status": 200
}
```

For get:

```
{
    "type":"get_outputgroup",
    "status":200,
    "value":[
        {"index":id1, "value":"id1 group value"},
        {"index":id2, "value":"id2 group value"},
        ... ...
        {"index":idx, "value":"idx group value"}
    ]
}
```

3.16 Get EQ Settings

3.16.1 Request

```
{"type":"get_eqinfo"}
```

3.16.2 Response

```
{
  "type": "get_eqinfo" ,
  "status": "200",
  "value ":[
    {
      "index ": dsp preset item index ,
      "value":
        [
          {
            "uuid":"xxx",
            "eq_enable":"0 or 1",
            "eq_freq":"20~20000",
            "eq_qratio":"0.3~24",
            "eq_gain":"-12~+12"
          },
          ...  ...
        ]
    },
    {
      "index ": dsp preset item index ,
      "value":
        [
          {
            "uuid":"xxx",
            "eq_enable":"0 or 1",
            "eq_freq":"20~20000",
            "eq_qratio":"0.3~24",
            "eq_gain":"-12~+12"
          }
          ...  ...
        ]
    }
  ]
}
```

The uuid is a GUID to indicate an unique EQ setting item.

The DSP preset item definition can be found in Table-1

3.17 Set EQ Settings

3.17.1 Request

```
{
  "type": "set_eqinfo",
  "value ":[
      {
        "index": dsp preset item index ,
        "value":
        [
          {"uuid":"xxx",
          "eq_enable":"0 or 1",
          "eq_freq":"20~20000",
          "eq_qratio":"0.3~24",
          "eq_gain":"-12~+12"}
        ]
      }
  ]
}
```

The DSP preset item definition can be found in Table-1

3.17.2 Response

```
{
  "type":"set_eqinfo" , "status":200
}
```

3.18 Add EQ Settings

3.18.1 Request

This is creating new EQ setting, which can create one or more eq items:

```
{
  "type": "add_eq" ,
  "value": [
     {"index ": dsp preset item index},
     ...   ...   ...  ...  ...  ...
     {"index ": dsp preset item index}
  ]
}
```

Suppose we want to add 2 EQ items:

Example 7

Creating 2 EQ items

3.18.2 Response

```
{ "type":"add_eq" , "status":200}
```

3.19 Delete EQ Settings

3.19.1 Requst

Suppose client wants to delete a eq uuid EQ item:

```
{
  "type":"delete_eq" ,
  "value":[
    {"index": dsp preset item index , "value": "eq uuid"}
  ]
}
```

3.19.2 Response

```
{"type":"delete_eq" , "status":200}
```

3.20 Low Tilt Per Output

3.20.1 Request

Set Enable:

```
{
    "type":"set_lowtiltenable" ,
    "value":[
       {"index": dsp preset item index , "value": "0 or 1"}
    ]
}
```

Get Enable:

```
{
    "type":"get_lowtiltenable",
    "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

The value defintion:

value	Meaning
"0"	off
"1"	on

Set Frequency:

```
{
  "type":"set_lowtiltfreq" ,
  "value":[
    {"index": dsp preset item index , "value": "20-20000"}
  ]
}
```

Get Frequency:

```
{
    "type":"get_lowtiltfreq",
    "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

Set Gain:

```
{
  "type":"set_lowtiltgain" ,
  "value":[
    {"index": dsp preset item index , "value": "-12~12"}
  ]
}
```

Get Gain:

```
{
    "type":"get_lowtiltgain",
    "status":200,
    "value":[
        {"index":dsppresetitemindex0, "value":"-12~12"},
        {"index":dsppresetitemindex1, "value":"-12~12"},
        ... ...
        {"index":dsppresetitemindexx, "value":"-12~12"}
    ]
}
```

3.20.2 Response

For Set Enable:

```
{
    "type":"set_lowtiltenable" ,
    "status":200
}
```

For Get Enable:

```
{
  "type":"get_lowtiltenable",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

For Set Frequency:

```
{
  "type":"set_lowtiltfreq" ,
  "status": 200
}
```

For Get Frequency:

```
{
    "type":"get_lowtiltfreq",
    "status":200,
    "value":[
        {"index":dsppresetitemindex0, "value":"20 -20000"},
        {"index":dsppresetitemindex1, "value":"20 -20000"},
        ... ...
        {"index":dsppresetitemindexx, "value":"20 -20000"}
    ]
}
```

For Set Gain:

```
{
  "type":"set_lowtiltgain" ,
  "status": 200
}
```

For Get Gain:

```
{
  "type":"get_lowtiltgain",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"-12~12"},
    {"index":dsppresetitemindex1, "value":"-12~12"},
    ... ...
    {"index":dsppresetitemindexx, "value":"-12~12"}
  ]
}
```

3.21 High Tilt Per Output

3.21.1 Request

Set Enable:

```
{
    "type":"set_hightiltenable" ,
    "value":[
       {"index": dsp preset item index , "value": "0 or 1"}
    ]
}
```

Get Enable:

```
{
  "type":"get_hightiltenable" ,
  "value":[ dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx ]
}
```

The value defintion:

value	Meaning
"0"	off
"1"	on

Set Frequency:

```
{
  "type":"set_hightiltfreq" ,
  "value":[
    {"index": dsp preset item index , "value": "20-20000"}
  ]
}
```

Get Frequency:

```
{
  "type":"get_hightiltfreq",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

Set Gain:

```
{
  "type":"set_hightiltgain" ,
  "value":[
    {"index": dsp preset item index , "value": "-12~12"}
  ]
}
```

Get Gain:

```
{
  "type":"get_hightiltgain",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

3.21.2 Response

For Set Enable:

```
{
    "type":"set_hightiltenable" ,
    "status":200
}
```

For Get Enable:

```
{
    "type":"get_hightiltenable",
    "status":200,
    "value":[
        {"index":dsppresetitemindex0, "value":"0 or 1"},
        {"index":dsppresetitemindex1, "value":"0 or 1"},
        ... ...
        {"index":dsppresetitemindexx, "value":"0 or 1"}
    ]
}
```

For Set Frequency:

```
{
  "type":"set_hightiltfreq" ,
  "status": 200
}
```

For Get Frequency:

```
{
    "type":"get_hightiltfreq",
    "status":200,
    "value":[
        {"index":dsppresetitemindex0, "value":"20 -20000"},
        {"index":dsppresetitemindex1, "value":"20 -20000"},
        ... ...
        {"index":dsppresetitemindexx, "value":"20 -20000"}
    ]
}
```

For Set Gain:

```
{
  "type":"set_hightiltgain" ,
  "status": 200
}
```

For Get Gain:

```
{
    "type":"get_hightiltgain",
    "status":200,
    "value":[
        {"index":dsppresetitemindex0, "value":"-12~12"},
        {"index":dsppresetitemindex1, "value":"-12~12"},
        ... ...
        {"index":dsppresetitemindexx, "value":"-12~12"}
    ]
}
```

4 Local Configuration Tool Only

4.1 Factory Reset

4.1.1 Request

```
{ "type":"device_factoryreset"}
```

4.1.2 Response

There would be a system power cycle after executing Factory Reset, so it's N/A for response data.

4.2 Reboot

4.2.1 Request

```
{"type":"device_reboot"}
```

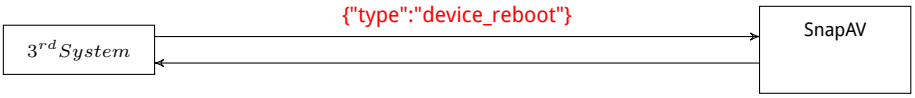
4.2.2 Response

N/A

As reboot would cause the whole system to restart, response data is N/A for this case.

Example 8

Reboot Device



4.3 Auto-On Method Abandon

4.3.1 Request

Set:

```
{
  "type":"set_automode" ,
  "value": "auto mode value"
}
```

Get:

```
{"type":"get_automode"}
```

The definition:

value	Meaning
"0"	Power Button
"1"	Voltage Trigger
"2"	Audio

4.3.2 Response

For Set:

```
{
  "type":"set_automode" ,
  "status": 200
}
```

For Get:

```
{
    "type":"get_automode",
    "status":200,
    "value":"auto mode"
}
```

4.4 Audio Sense Method

4.4.1 Request

Set:

```
{
  "type":"set_audiosense" ,
  "value": "0 or 1"
}
```

Get:

```
{"type":"get_audiosense"}
```

The definition:

value	Meaning
"0"	disable
"1"	enable

4.4.2 Response

For Set:

```
{
  "type":"set_audiosense" ,
  "status": 200
}
```

For Get:

```
{
    "type":"get_audiosense",
    "status":200,
    "value":"audio mode"
}
```

4.5 Power Button Method

4.5.1 Request

Set:

```
{
  "type":"set_powerbutton" ,
  "value": "0 or 1"
}
```

Get:

```
{"type":"get_powerbutton"}
```

The definition:

value	Meaning
"0"	disable
"1"	enable

4.5.2 Response

For Set:

```
{
  "type":"set_powerbutton" ,
  "status": 200
}
```

For Get:

```
{
    "type":"get_powerbutton",
    "status":200,
    "value":"powerbutton mode"
}
```

4.6 Sleep Mode

4.6.1 Request

Set:

```
{
  "type":"set_sleepmode" ,
  "value ":[
    {"index": id , "value": "sleep mode value"}
  ]
}
```

Get:

```
{
    "type":"get_sleepmode",
    "value":[id1, id2, ..., idx]
}
```

value	Meaning
"0"	After 5 Min
"1"	After 15 Min
"2"	After 30 Min
"3"	After 1 hour
"4"	After 3 hour

4.6.2 Response

For Set:

```
{
  "type":"set_sleepmode" ,
  "status":200
}
```

For Get:

```
{
    "type":"get_sleepmode",
    "status":200,
    "value":[
        {"index":id1, "value":"id1 sleep mode value"},
        {"index":id2, "value":"id2 sleep mode value"},
        ... ...
        {"index":idx, "value":"idx sleep mode value"}
    ]
}
```

4.7 Mode Source2

4.7.1 Request

For Set:

```
{
  "type":"set_outputsource2mode" ,
  "value ":[
    {"index": id , "value": "mode value"}
  ]
}
```

For Get:

```
{
    "type":"get_outputsource2mode",
    "value":[id1, id2, ..., idx]
}
```

The definition of value:

value	Meaning
"0"	off
"1"	mix
"2"	mute

4.7.2 Response

For Set:

```
{
  "type":"set_outputsource2mode",
  "status": 200
}
```

For Get:

```
{
    "type":"get_outputsource2mode",
    "status":200,
    "value":[
        {"index":id1, "value":"id1 source2 mode"},
        {"index":id2, "value":"id2 source2 mode"},
        ... ...
        {"index":idx, "value":"idx source2 mode"}
    ]
}
```

4.8 Gain Offset

4.8.1 Request

Set:

```
{
  "type":"set_gainoffset" ,
  "value ":[
    {"index ": id , "value": "gain offset value"}
```

```
  ]
}
```

Get:

```
{
    "type":"get_gainoffset",
    "value":[id1, id2, ..., idx]
}
```

The value ranges in $[-10, 10]$

4.8.2 Response

For Set:

```
{
  "type":"set_gainoffset" ,
  "status": 200
}
```

For Get:

```
{
    "type":"get_gainoffset",
    "status":200,
    "value":[
        {"index":id1, "value":"-10 ~ 10"},
        {"index":id2, "value":"-10 ~ 10"},
        ... ...
        {"index":idx, "value":"-10 ~ 10"}
    ]
}
```

4.9 Test Signal

4.9.1 Request

For Set:

```
{
  "type":"set_testsignal",
  "value":[
     {"index":id, "testenable":"0 or 1","testsignal":"0","testvol":"vol value"}
  ]
}
```

For Get:

```
{
    "type":"get_testsignal",
    "value":[id1, id2, ..., idx]
}
```

4.9.2 Response

For Set:

```
{
  "type":"set_testsignal", "status":200
}
```

For Get:

```
{
  "type":"get_testsignal",
  "status":200,
  "value":[
    {"index":id1, "testenable":"0 or 1", "testsignal":"id1 input index", "testvol":"id1 volume value"},
    {"index":id2, "testenable":"0 or 1", "testsignal":"id2 input index", "testvol":"id2 volume value"},
    ... ...
    {"index":idx, "testenable":"0 or 1", "testsignal":"idx input index", "testvol":"idx volume value"}
  ]
}
```

4.10 Low Pass

4.10.1 Requeset

Set Enable:

```
{
  "type":"set_lowpassenable" ,
  "value":[
    {"index": dsp preset item index , "value": "0 or 1"}
  ]
}
```

Get Enable:

```
{
    "type":"get_lowpassenable",
    "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

The value defintion:

value	Meaning
"0"	off
"1"	on

Set Frequency:

```
{
  "type":"set_lowpassfreq" ,
  "value":[
    {"index": dsp preset item index , "value": "20-20000"}
  ]
}
```

Get Frequency:

```
{
  "type":"get_lowpassfreq",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

Set Filter:

```
{
  "type":"set_lowpassfilter" ,
  "value ":[
    {"index ": dsp preset item index , "value": "filter value"}
  ]
}
```

Get Filter:

```
{
    "type":"get_lowpassfilter",
    "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

Filter value: $6, 12, 18, 24$

4.10.2 Response

For Set Enable:

```
{
  "type":"set_lowpassenable" ,
  "status":200
}
```

For Get Enable:

```
{
  "type":"get_lowpassenable",
  "status":200,
  "value":[
        {"index":dsppresetitemindex0, "value":"0 or 1"},
        {"index":dsppresetitemindex1, "value":"0 or 1"},
        ... ...
        {"index":dsppresetitemindexx, "value":"0 or 1"}
  ]
}
```

For Set Frequency:

```
{
  "type":"set_lowpassfreq" ,
  "status":200
}
```

For Get Frequency:

```
{
  "type":"get_lowpassfreq",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"20 ~ 20000"},
    {"index":dsppresetitemindex1, "value":"20 ~ 20000"},
    ... ...
    {"index":dsppresetitemindexx, "value":"20 ~ 20000"}
  ]
}
```

For Set Filter:

```
{
  "type":"set_lowpassfilter" ,
  "status":200
}
```

For Get Filter:

```
{
    "type":"get_lowpassfilter",
    "status":200,
    "value":[
        {"index":dsppresetitemindex0, "value":"6/12/18/24"},
        {"index":dsppresetitemindex1, "value":"6/12/18/24"},
        ... ...
        {"index":dsppresetitemindexx, "value":"6/12/18/24"}
    ]
}
```

4.11 High Pass

4.11.1 Request

Set Enable:

```
{
  "type":"set_highpassenable" ,
  "value ":[
    {"index": dsp preset item index , "value": "0 or 1"}
  ]
}
```

Get Enable:

```
{
  "type":"get_highpassenable",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

value	Meaning
"0"	off
"1"	on

Set Frequency:

```
{
  "type":"set_highpassfreq" ,
  "value":[
    {"index": dsp preset item index , "value": "20-20000"}
  ]
}
```

Get Frequency:

```
{
  "type":"get_hightiltfreq",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

Set Filter:

```
{
  "type":"set_highpassfilter" ,
  "value":[
    {"index ": dsp preset item index , "value": "filter value"}
  ]
}
```

Get Filter:

```
{
  "type":"get_highpassfilter",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

Filter value: $6, 12, 18, 24$

4.11.2 Response

For Set Enable:

```
{
  "type":"set_highpassenable" ,
  "status":200
}
```

For Get Enable:

```
{
  "type":"get_highpassenable",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"0 or 1"},
    {"index":dsppresetitemindex1, "value":"0 or 1"},
    ... ...
    {"index":dsppresetitemindexx, "value":"0 or 1"}
  ]
}
```

For Set Frequency:

```
{
  "type":"set_highpassfreq" ,
  "status":200
}
```

For Get Frequency:

```
{
  "type":"get_hightiltfreq",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"20 ~ 20000"},
    {"index":dsppresetitemindex1, "value":"20 ~ 20000"},
    ... ...
    {"index":dsppresetitemindexx, "value":"20 ~ 20000"}
  ]
}
```

For Set Filter:

```
{
  "type":"set_highpassfilter" ,
  "status":200
}
```

For Get Filter:

```
{
  "type":"get_highpassfilter",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"6/12/18/24"},
```

```
    {"index":dsppresetitemindex1, "value":"6/12/18/24"},
    ... ...
    {"index":dsppresetitemindexx, "value":"6/12/18/24"}
  ]
}
```

4.12 Delay

4.12.1 Request

Set Msec:

```
{
  "type":"set_delayms" ,
  "value ":[
    {"index": dsp preset item index , "value": "ms"}
  ]
}
```

Get Msec:

```
{
  "type":"get_delayms",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

The value range is in $[0, 12]$, can be in float format.

Set Feet:

```
{
  "type":"set_delayfeet" ,
  "value":[
    {"index": dsp preset item index , "value": "feet value"}
  ]
}
```

Get Feet:

```
{
  "type":"get_delayfeet",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

Set Meter:

```
{
  "type":"set_delaymeter" ,
  "value":[
    {"index": dsp preset item index , "value": "meter value"}
  ]
}
```

Get Meter:

```
{
  "type":"get_delaymeter",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

4.12.2 Response

For Set Msec:

```
{
  "type":"set_delayms" ,
  "status":200
}
```

For Get Msec:

```
{
  "type":"get_delayms",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"0 ~ 12.00"},
    {"index":dsppresetitemindex1, "value":"0 ~ 12.00"},
    ... ...
    {"index":dsppresetitemindexx, "value":"0 ~ 12.00"}
  ]
}
```

For Set Feet:

```
{
  "type":"set_delayfeet" ,
  "status":200
}
```

For Get Feet:

```
{
  "type":"get_delayfeet",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"feet value"},
    {"index":dsppresetitemindex1, "value":"feet value"},
    ... ...
    {"index":dsppresetitemindexx, "value":"feet value"}
  ]
}
```

For Set Meter:

```
{
  "type":"set_delaymeter" ,
  "status":200
}
```

For Get Meter:

```
{
  "type":"get_delaymeter",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"meter value"},
    {"index":dsppresetitemindex1, "value":"meter value"},
    ... ...
    {"index":dsppresetitemindexx, "value":"meter value"}
  ]
}
```

4.13 Limiter

4.13.1 Request

Set:

```
{
  "type":"set_limiter" ,
  "value":[
    {"index": dsp preset item index , "value": "limiter value"}
  ]
}
```

Get:

```
{
  "type":"get_limiter",
  "value":[dsppresetitemindex0, dsppresetitemindex1, ..., dsppresetitemindexx]
}
```

The value : $[0, -1, -2, -3, -4, -5]$

4.13.2 Response

For Set:

```
{
  "type":"set_limiter",
  "status":200
}
```

For Get:

```
{
  "type":"get_limiter",
  "status":200,
  "value":[
    {"index":dsppresetitemindex0, "value":"-5 ~ 0"},
    {"index":dsppresetitemindex1, "value":"-5 ~ 0"},
    ... ...
    {"index":dsppresetitemindexx, "value":"-5 ~ 0"}
  ]
}
```

4.14 Copy DSP Preset

4.14.1 Request

```
{
  "type":"copy_dsp",
  "value":[
    {"from": "dsp preset item index" , "to": "dsp preset item index"}
  ]
}
```

The DSP preset item definition can be found in Table-1

4.14.2 Response

```
{
  "type":"copy_dsp" , "status":200
}
```

5 Unsolicited Message

If there's any status changes, AMP device would send Unsolicited message to the client.

By convention, all these reported status are the type with `get_` prefix, and the data layout is the same as the GET cases.

Client just gets these message directly, without any sending any fetch message at all.

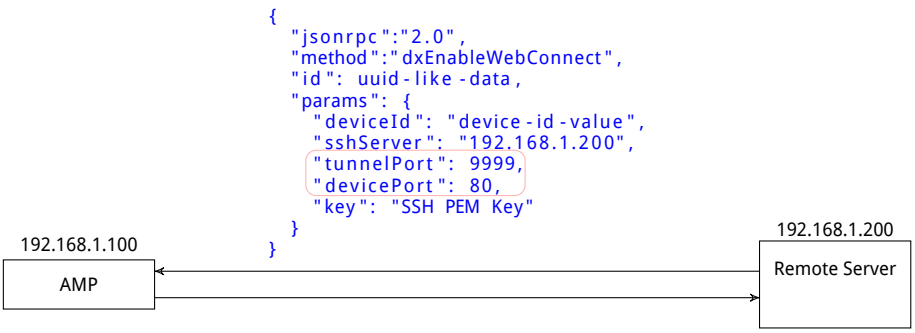
Appendices

A Access Local WEB UI HOWTO

Here's description for Hansong's local web UI accessing feature, especially for Issue report from [3].

Example 9

Flow Chart of the Local WEB UI Accessing

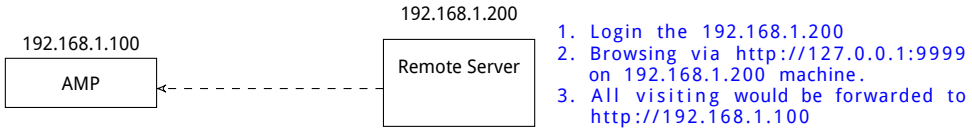


After the connection established, all visiting to remote server would go to the target AMP unit.

But keep in mind that this is happended on the localhost(i.e, 127.0.0.1) for such accessing action by default.

Example 10

Accessing the AMP web via Remote Server



B Cheat Sheet

B.1 Avoid Port Confliction

Before setup the reverse channel like above Example, user MUST make sure the remote server's port is not occupied.

For example, suppose the remote server(192.168.1.200 for above Example) already launched a webserver(such as Apache, or Nginx).

Then the HTTP port 80 is already in-use, and we would failed setup the 80 on the remote server due to port confliction:

```
{
  "jsonrpc":"2.0",
  "method":"dxEnableWebConnect",
  "id": uuid-like-data,
  "params": {
    "deviceId": "device-id-value",
    "tunnelPort": 80,
    "devicePort": 80,
    "key": "SSH PEM Key"
  }
}
```

B.2 http://192.168.1.200 instead of http://127.0.0.1

The reverse channel setup by SSH won't enable the wildcard IP address(0.0.0.0) by default, only local IP address(127.0.0.1) is enabled.

This policy is taken by the open source community due to security consideration.

Suppose user wants to setup an N mapping-port to target AMP unit:

```
{
  "jsonrpc":"2.0",
  "method":"dxEnableWebConnect",
  "id": uuid-like-data,
  "params": {
    "deviceId": "device-id-value",
    "tunnelPort": N,
    "devicePort": 80,
    "key": "SSH PEM Key"
  }
}
```

Copyright ©2021 Wirepath Home Systems, LLC. All rights reserved. Control4 and SnapAV and their respective logos are registered trademarks or trademarks of Wirepath Home Systems, LLC, dba “Control4” and/or dba “SnapAV” in the United States and/or other countries. 4Store, 4Sight, Control4 My Home, Snap AV, Araknis Networks, BakPak, Binary, Dragonfly, Episode, Luma, Mockupancy, Nearus, NEEO, Optiview, OvrC, Pakedge, Sense, Strong, Strong Evolve, Strong Versabox, SunBriteDS, SunBriteTV, Triad, Truvision, Visualint, WattBox, Wirepath, and Wirepath ONE are also registered trademarks or trademarks of Wirepath Home Systems, LLC. Other names and brands may be claimed as the property of their respective owners. All specifications subject to change without notice.

Rev 200-00692 Response-Amp-API-A 210412 TW