Netvio Universal API Version 1.0

Revision	1.0.4
Written By	James Meredith & Nick Roberts
Date	07/06/2023

NETVIO

Table of Contents

1 Disclaimer

The information contained in this manual does not guarantee compatibility or operability of Netvio models with all other equipment and systems. Netvio is not responsible for product malfunctions resulting from failure to follow the instructions and information contained herein.
The information and specifications contained herein are subject to change without notice.

2 Overview

This API is a specification for how a client should request that resources be fetched or modified, and how a server should respond to those requests used in serial communication (RS-232C) and network communication (TCP/IP) for Netvio Systems.

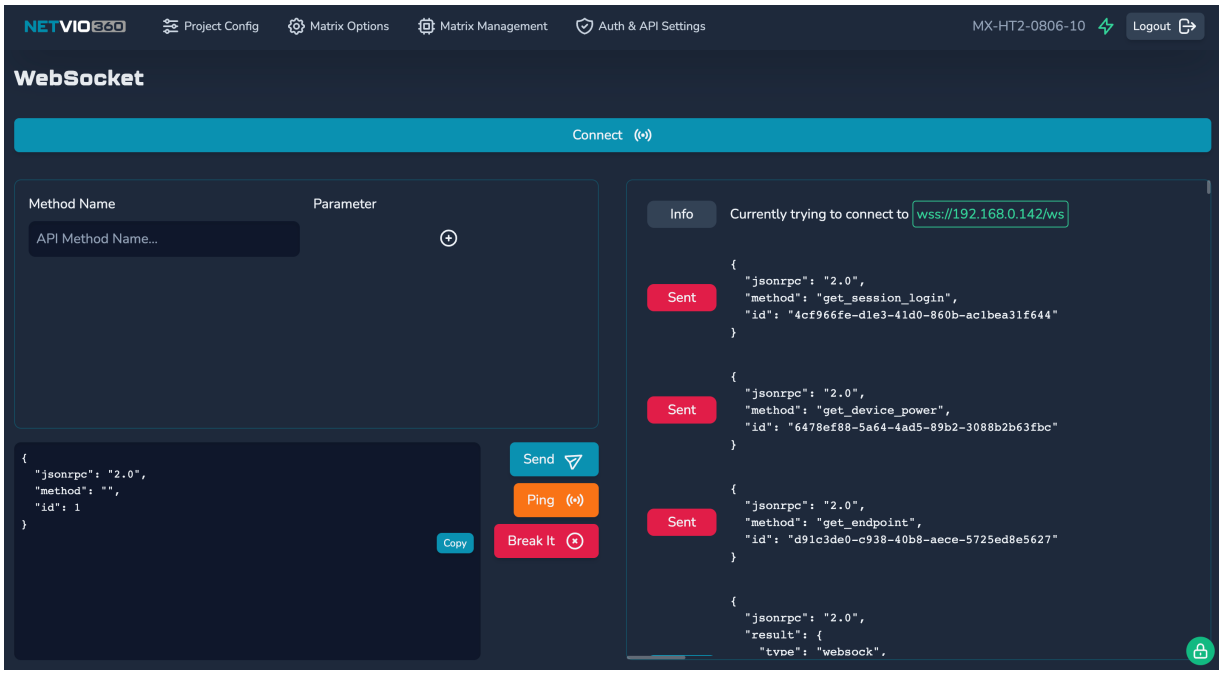
The API is designed to minimize both the number of requests and the amount of data transmitted between clients and servers. This efficiency is achieved without compromising readability, flexibility, or discoverability.

3 Netvio Developer Interface

The Netvio web UI has a hidden development interface to assist developers who are creating Netvio control drivers. All Netvio matrix and AVoIP solutions support this interface that enables both the viewing and testing of API communications.

To launch the Netvio Developer Interface, log in to either the Netvio matrix or CL controller web UI. Either through a browser by using the URL https://IPADDRESS, or through the Netvio 360 configuration software. After entering your username and password you will see the Project Config page.

- To enter the Developer interface on a Windows device press Command and Enter together.
- To enter the Developer interface on a MacOS device press Control and Enter together.



4 Session Establishment

4.0.0 Websocket Secure (WSS)

The secure API session can be established using WSS. The Websocket Protocol is detailed in RFC6455.

Communication uses the API Server HTTPS/WSS TCP port #**443**. HTTPS and WSS connections are therefore tunnelled over TLS.
TLS connections will use the API server's TLS host certificate that has been setup by the user in advance of using the Driver. Note, that this could include the default, auto-generated certificate.
TLS connections may also be required to use Client-side certificates if this feature has been setup by the user in advance of using the Driver.

The WSS connection can be established by sending the standard Websocket client handshake HTTPS **GET** request to the API server:

```
wss://<IP:PORT>/ws
```

Where *<IP:PORT>* is the IPv4 address and optionally the TCP port of the API server, e.g. `192.168.3.45:443`.

The HTTPS request must include the following Header key and value pairs:

```
host: <HOST>
upgrade: websocket
connection: upgrade
sec-websocket-version: 13
sec-websocket-key: <system generated key>
sec-websocket-protocol: netvio-websocket
```

Where *<HOST>* is the IPv4 address & TCP port of the API server, e.g. `192.168.3.45:443`.
Where *<system generated key>* is a [random] base-64 encoded nonce generated by the Websocket client application.

Further detail can be obtained by referencing RFC6455.

4.1.0 TCP Socket

The plaintext API session can be established using a raw TCP socket connection.
TCP is detailed in RFC793.
Communication uses the API Server TCP port #**23**.
It must be noted that this connection is not Telnet and does not support Telnet-specific option requests or control functions.

4.2.0 TCP Socket String Delimiters

Once the TCP socket is established, all command strings must be terminated with \n (hex byte 0A). Commands may also be delimited by \r\n (hex bytes 0D 0A).

Responses and notifications from the server will be terminated with \r\n (hex bytes 0D 0A).

4.3.0 JSON-RPC

The API is based on the JSON-RPC 2.0 stateless, light-weight remote procedure call protocol. The JSON-RPC 2.0 Specification can be referenced at https://www.jsonrpc.org/specification
Primarily this specification defines several data structures and the rules around their processing.

For commands sent from the client to the server, the command is indicated by using a json-rpc request object with the appropriate value for the mandatory json-rpc **method** key. Additional command parameters are contained within the value of the optional json-rpc **params** key.

For the API, a request object always includes a mandatory **id** key/value pair. The value is not defined by the API and can be any valid value to be used by the Client application (e.g. for process tracking).

For the API, json-rpc notifications (a request object with no **id** key) is never used from client to server.

The server will always answer a request object sent from the client with a response object. The **id** key will match that of the associated request object.

In addition, the server may respond with a json-rpc error response object if the json-rpc call encounters an error.

For asynchronous notifications from server to client, the server will send a json-rpc notification request object (with no **id** key) to the client. The client is not expected to respond directly to this notification.

Please note that problems can arise when copy/pasting commands directly from word and PDF documents as the ASCII representations may be misinterpreted resulting in a command that looks correct but uses incorrect hex representations of ASCII characters. This is particularly the case with apostrophes (“ ”) which are prevalent throughout this API.

5 Serial Communication (RS-232C)

5.0.0 Cable Length

Guaranteed cable length: 15 m (However, the cable length may not be able to be guaranteed depending on the cable used.)
A connection is used only TxD and RxD lines.

5.1.0 Communication Specifications

Full duplex communication channel Asynchronous system
No flow control
Transfer rate: 115,200 bps

The bit configuration is as follows:

1 start bit + 8-data bits + No Parity + 1 stop bit

6 API Objects

The API utilises the concept of JSON objects when structuring API commands and responses. The key names and values used within a **method** pertain to an API object, usually though the key **`"target":`**, although refer to the Universal API Command Detail document to evaluate the correct command syntax. Note that for the majority of commands, if the key **`"target":`** is omitted the target object will be a set default value.

A description of some of the relevant API objects and their keys are listed below:

6.0.0 Endpoints

An endpoint is the name given to a virtual API object that closely represents a hardware input subsystem or output subsystem of a matrix switch. For example, Netvio matrix switchers MX-H2-0404-10 and MX-HT2-0x0x-10 have a fixed number of input endpoints and a fixed number of output endpoints.

Endpoints can be referenced in the API using the JSON key **`"endpointID":`** or **`"target":`** where the value is a set format. The format includes whether the endpoint is an input, output or controller and the index of the endpoint.

For example, the valid values for **`"endpointID":`** on the MX-HT2-0404 are as follows:

`"endpointID": "in1"` Input 1
`"endpointID": "in2"` Input 2
`"endpointID": "in3"` Input 3
`"endpointID": "in4"` Input 4

"endpointID": "out1"	Output 1
"endpointID": "out2"	Output 2
"endpointID": "out3"	Output 3
"endpointID": "out4"	Output 4
"endpointID": "cl1"	Internal Controller

The endpoint object is often referenced directly (or by a default if omitted) within the JSON-RPC call by using the fixed value format above, for the **"endpointID":** or **"target":** keys.

Please see section 7 for a list of relevant endpoints for each Netvio model.

6.1.0 Signals

An input or output endpoint object includes a set of nested objects referred to as signal nodes. A signal is the name given to a virtual data stream that closely represents a media or control signal. A signal egresses an endpoint towards the internal backplane/crosspoint switch (matrix switch) or via Ethernet to the network switch (AV over IP) and is then routed to the ingress of another endpoint.

Signals types include:

- Video
- Audio
- Infrared
- Serial (AVoIP Only)
- USB (AVoIP Only)

The point at which a signal ingresses or egresses an endpoint is the signal node.

Signal nodes can be referenced in the API using the JSON key **"***signalNode***":** where the value is a set format. The format includes the type of signal, whether the signal node is an input or output and the index of the signal (as there could be more than one of the same type).

For example, the valid values for **"***signalNode***":** on the MX-HT2-0404 nested within the input object identified by **"***endpointID***": "***out1***"** are as follows:

"signalNode": "out1-videorx1"	Video signal input to the HDMI/HDbT output
"signalNode": "out1-audiorx1"	Audio signal input to the HDMI/HDbT output
"signalNode": "out1-audiorx2"	Audio signal input to the L/R & S/PDIF output
"signalNode": "out1-audiotx1"	Audio signal output from HDMI ARC (if enabled)
"signalNode": "out1-audiotx2"	Audio signal output from HDbaseT audio return
"signalNode": "out1-infraredtx1"	Infrared signal output from HDBaseT extender

Please see section 8 for a list of relevant signal nodes for each Netvio model.

7 Netvio Universal API Commands

7.0.0 Initiate Session/Session Login

set_session_login

Required for establishing API session login. Applies when a new TCP socket connection is established with the Controller or the session has timed-out due to inactivity (if timeout is enabled). It is not required when using the Websocket secure session method for new connections and only applies if the session user is logged out.

On non-persistent client connections, if the control system does not support prefixing login commands, then a macro should be created to send the initiation followed by the relevant command.

Example Command

```
{"jsonrpc": "2.0", "method": "set_session_login", "params": {"username": "admin", "pw": "password"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Params Key	Values	Description	Example
username	ascii(4 to 31 char.)	Username	"admin"
password	ascii(8 to 63 char.)	Password	"password"
cmdStatus	"success"	Command received by API server	"success"

7.1.0 End Session/Session Logout

set_session_logout

Logs out a User from the current API session.

Although the Netvio product will allow simultaneous connection of hundreds of active sessions, it is best practice to logout of each session. Particularly if the cyctem does not handle active connection sessions and instead only supports one shot commands.

In this case create a macro to: Login > Send Commnd > Logout

Example Command

{"jsonrpc": "2.0", "method": "set_session_logout", "id": 1}

7.2.0 Signal Switching

set_matrix_assign

Sets internal matrix switch assignments of endpoint signals. Signals are the internally routed audio, video and control signals, through the system.

Signal inputs will assign to output nodes. Signal output nodes will also assign to input nodes. Signal input nodes will not assign to other input nodes as signal output nodes will not assign to other output nodes.

If a "target" Endpoint is specified, the assignment will attempt to assign all "target" Endpoint signal nodes. If an "assign" Endpoint is specified, the assignment will attempt to assign all "assign" Endpoint signal nodes.

The command can either:
1. Assign all Endpoint signals inputs to another Endpoint's equivalent signals outputs.
2. Assign a specific Endpoint signal node to another Endpoint.

Example Command

Assign output endpoint “out3” to input endpoint “in6”:

```
{"jsonrpc": "2.0", "method": "set_matrix_assign", "params": {"target": "out3", "assign": "in6"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Example Command

Assign signal node “out3-videorx1” to signal node “in6-videotx1”:

```
{"jsonrpc": "2.0", "method": "set_matrix_assign", "params": {"signalNode": "out3-videorx1", "assignSignal": "in6-videotx1"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Params Key	Values	Description	Example 1
target	endpoint	Valid targets include any project Endpoint	"in1"
signalNode	signal	Valid signals include any project Endpoint's signal node	"out1-videorx1"
assign	endpoint	Valid assign include any project Endpoint	"out4"
assignSignal	signal	Valid signals include any project Endpoint's signal node	"in4-audiotx1"
cmdStatus	"success"	Command received by API server	"success"

7.3.0 Signal Status

get_matrix_assign

To obtain the current state of discrete signals assignments and optionally extrapolate the state of endpoint assignments.

Example Command

Get current assignment of signal node “out3-videorx1”:

```
{"jsonrpc": "2.0", "method": "get_matrix_assign", "params": {"signalNode": "out3-videorx1"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"signalNode": "out3-videorx1", "assignSignal": "in4-videotx1"}, "id": 1}
```

Example Command

Get current assignment of all signal nodes for the IP-JP4-CL-10 endpoint “out3”:

```
{"jsonrpc": "2.0", "method": "get_matrix_assign", "params": {"target": "out1"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"endpointID": "out3", "signals": [{"signalNode": "out3-videorx1", "assignSignal": "in6-videotx1"}, {"signalNode": "out3-audiorx1", "assignSignal": "in3-audiotx1"}, {"signalNode": "out3-infraredrx1", "assignSignal": "in3-infraredtx1}, {"signalNode": "out3-serialrx1", "assignSignal": "in3-serialtx1"}, {"signalNode": "out3-infraredtx1", "assignSignal": "in3-infraredrx1"}, {"signalNode": "out3-serialtx1", "assignSignal": "in3-serialrx1"}]}, "id": 1}
```

7.4.0 Signal Status notifications

notify_matrix_assign

An asynchronous event sent from the Controller notifying the session client that the discrete signals assignment has changed.

Example Response

Front panel buttons change a Matrix Switch signal node assignment for the endpoint “out1”.

```
{"jsonrpc": "2.0", "method": "notify_matrix_assign", "params": {"endpointID":
"out1", "signals": [{"signalNode": "out1-videorx1", "assignSignal": "in3-
videotx1"}, {"signalNode": "out1-audiorx1", "assignSignal": "in3-audiotx1"},
{"signalNode": "out1-audiorx2", "assignSignal": "in3-audiotx1}, {"signalNode":
"out1-audiotx1", "assignSignal": null}, {"signalNode": "out1-audiotx2",
"assignSignal": "out4-audiorx2"}, {"signalNode": "out1-infraredtx1",
"assignSignal": "in3-infraredrx1"}]}}
```

7.5.0 Signal Information

get_video_details

To obtain current video timing information for a video signal on an endpoint.

Example Command

Get current video signal details of Matrix Switch signal node “in2-videotx1”:

```
{"jsonrpc": "2.0", "method": "get_video_details", "params": {"target": "in3"},
"id": 1}
```

Example Response

```
{
  "jsonrpc": "2.0",
  "result": {
    "endpointID": "in3",
    "signals": [
      {
        "type": "video",
        "map": 1,
        "signalNode": "in3-videotx1",
        "assignSignal": null,
        "availPorts": [
          "HDMI_IN_1"
        ],
        "srcPort": "",
        "direction": "output",
        "inVideoResolution": "1920x1080",
        "inVideoFPS": 60
      }
    ]
  },
  "id": 1
}
```

7.6.0 Set Audio Volume

set_audio_volume

Sets audio output volume for the analog audio output on the defined Endpoint(s)

Example Command

Set audio volume at 22 for the analog output on endpoint “out1”:

```
{"jsonrpc": "2.0", "method": "set_audio_volume", "params": {"target": "out1",
"audioOutVolume": 22}, "id": 1}
Example Response
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Example Command

Increase audio volume by 5 for the analog output on endpoint “out3”:

```
{"jsonrpc": "2.0", "method": "set_audio_volume", "params": {"target": "out3",
"audioOutVolume": "+5"}, "id": 1}
Example Response
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Example Command

```
{"jsonrpc": "2.0", "method": "set_audio_volume", "params": {"target": "out3",
"audioOutMute": true}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Example Command

Mute the analog output on endpoint “out3”:

```
{"jsonrpc": "2.0", "method": "set_audio_volume", "params": {"target": "out3",
"audioOutMute": true}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Params Key	Values	Description	Example 1	Example 2
target	endpoint	Valid targets include any project Endpoint or Group. If omitted the target defaults to the device running the API session	"in1"	
audioOutMute	true, false	The audio mute state true = audio is muted false = audio volume is audioOutVolume If omitted, the current audioOutMute will be unchanged	true	false
audioOutDelay	int(0, 300)	The audio delay in milliseconds (ms) If omitted, the current audioOutDelay will be unchanged	0	100
audioOutVolume	int(0, 30)	number = The absolute audio volume as a reference to the device's HW range	5	"+5"

		"+<value>" string = A relative increase by <value> steps "-<value>" string = A relative decrease by <value> steps where max value = 30, min value = 0 If omitted, the current audioOutVolume will be unchanged		
cmdStatus	"success"	Command received by API server	"success"	

7.7.0 Get Audio Volume

get_audio_volume

To obtain the current volume and mute settings for an analog audio output on an endpoint

Example Command

Get the current audio volume for the Matrix Switch analog output on endpoint “out3”:

```
{"jsonrpc": "2.0", "method": "get_audio_volume", "params": {"target": "out3"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"endpointID": "out3", "processID": 2, "processName":
"analogAudio", "audioOutVolume": 100, "audioOutMute": false, "audioOutDelay": 0}, "id": 1}
```

Example Command

Get the current audio volume for the IP-JP4 analog output on endpoint “out3”:

```
{"jsonrpc": "2.0", "method": "get_audio_volume", "params": {"target": "out3"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"endpointID": "out3", "processID": 2, "processName":
"analogAudio", "audioOutVolume": 100, "audioOutMute": false}, "id": 1}
```

7.8.0 Volume notifications

An asynchronous event sent from the Controller notifying the session client that the volume and mute settings have changed.

Example Response

Matrix Switcher:

```
{"jsonrpc": "2.0", "method": "notify_audio_volume", "params": {"endpointID": "out1",
"processID": 2, "processName": "analogAudio", "audioOutVolume": 100, "audioOutMute":
false, "audioOutDelay": 0}}
```

Example Response

IP-JP4:

```
{"jsonrpc": "2.0", "method": "notify_audio_volume", "params": {"endpointID": "out1",
"processID": 2, "processName": "analogAudio", "audioOutVolume": 100, "audioOutMute":
false}}
```

7.9.0 Send IR

send_ir

Generates an infrared signal from the defined endpoint(s). Infrared code string formats must be in Pronto HEX format.

Example Command

Send an IR command to out1:

```
{"jsonrpc": "2.0", "method": "send_ir", "params": {"target": "out1", "irData": "0000 0067 0000 000d 0060 0018 0018 0018 0030 0018 0018 0018 0018 0018 0030 0018 0018 0018 0018 0018 0030 0018 0018 0018 0018 0018 0018 0018 0018 041f"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Params Key	Values		Description	Example 1	Example 2
irData	ir_code		The infrared pulse chain, represented as a recognised command string format (Pronto Hex)	0000 0067 0000 000d 0060 0018 0018 0018 0030 0018 0018 0018 0018 0018 0030 0018 0018 0018 0018 0018 0030 0018 0018 0018 0018 0018 0018 0018 0018 041f	
target	endpoint	This CL	Valid targets include any project Endpoint	"out1"	
irRepeat	int(1, 32)	1	The number of times the IR data is immediately repeated. For GC and DP formats this will overwrite the existing repeat value in the format string	1	8
cmdStatus	"success"		Command received by API server	"success"	

7.10.0 Send Serial

send_serial

Sets the settings for the Serial port on the defined Endpoint(s). For matrix, there are two output Endpoint RS-232 ports, the HDBaseT port and the local ports.

Please note that the serial port behaviour is influenced by the Serial settings of the Netvio Device. Serial port configuration must be configured in the device's I/O serial settings for example.

Example Command

Send the ascii command command01 with carriage return and line feed to out1.

```
{"jsonrpc": "2.0", "method": "send_serial", "params": {"target": "out1", "serialCode": "ascii", "serialData": "command01\n\r"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Params Key Mandatory	Values		Description	Example 1	Example 2
serialCode	"ascii", "hex"		The coding of the bytes in serialData. The data will either be interpreted as an ASCII or hexadecimal string	"ascii"	"hex"
serialData	ascii_s(1, 256)		The serial string. If serialCode = ascii, the string can contain any printable ascii character. Use JSON escape notation (\", \\, \b, \f, \n, \r, \t, \u) if serialCode = hex the string can only contain space-separated hexadecimal bytes	"command01\n\r"	"43 6f 6d 6d 61 6e 64 30 31 0a 0d"
target	endpoint, group	This CL	Valid targets include any project Endpoint If omitted the target defaults to the device running the API session	"out1"	
Params Key Optional	**Values**	**Default**	**Description**	**Example 1**	**Example 2**
uartID	1, 2	1	The uartID For MX-HT2-0808: 1 = HDBaseT RS-232 port 2 = Local RS-232 port If omitted, uartID = 1	1	2
cmdStatus	"success"		Command received by API server	"success"	

7.11.0 Send CEC

send_cec

To generate a cec bus control signal from a video port on an endpoint

Example Command

Generate a CEC command from the HDMI port on the IP-JP4 endpoint “out1”:

```
{"jsonrpc": "2.0", "method": "send_cec", "params": {"target": "out1", "cecData": "F0 36"}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

7.12.0 Send Proxy Commands

send_proxy_command

Generates a proxy command from the defined endpoint(s). Proxy commands are unique Infrared, Serial or CEC commands for each endpoint for Power and Volume control.

Proxy commands can be useful for when global functions are required across multiple Endpoints where the command data is different, e.g. Power Off

Example Command

```
{"jsonrpc": "2.0", "method": "send_proxy_command", "params": {"target": "out1",
"proxyComRef": 1}, "id": 1}
```

Example Response

```
{"jsonrpc": "2.0", "result": {"cmdStatus": "success"}, "id": 1}
```

Params Key Mandatory	Values	Description	Example 1	Example 2
proxyComRef	int(1, 10)	The index reference for the proxy command	1	2
target	endpoint, group	Valid targets include any project Endpoint or Group. If omitted the target defaults to the device running the API session	"out1"	
Params Key Optional	**Values**	**Description**	**Example 1**	**Example 2**
cmdStatus	"success"	Command received by API server	"success"	

8 Endpoints

The driver shall use a fixed range of values used to reference endpoints using the **"*endpointID*":**, **"*target*":** or **"assign":** keys. This will depend on the product the driver is being used with, as follows:

8.0.0 Netvio IP-JP4-CL-10 Endpoint References

"endpointID": "inN" Input **N**
"endpointID": "outN" Output **N**
"endpointID": "cl1" AV over IP Controller

Where 1 ≤ N ≤ 128

8.1.0 Netvio MX-H2-0404-10 Endpoint References

"endpointID": "inN" Input **N**
"endpointID": "outN" Output **N**
"endpointID": "cl1" Internal Controller

Where 1 ≤ N ≤ 4

8.2.0 Netvio MX-HT2-0404-10 Endpoint References

"endpointID": "inN" Input **N**
"endpointID": "outN" Output **N**
"endpointID": "cl1" Internal Controller

Where 1 ≤ N ≤ 4

8.3.0 Netvio MX-HT2-0806-10 Endpoint References

"endpointID": "inN" Input **N**
"endpointID": "outN" Output **N**
"endpointID": "cl1" Internal Controller

Where 1 ≤ N ≤ 8

8.4.0 Netvio MX-HT2-0808-10 Endpoint References

"endpointID": "inN" Input **N**
"endpointID": "outN" Output **N**
"endpointID": "cl1" Internal Controller

Where 1 ≤ N ≤ 8

9 Signal Node References

The driver shall use a fixed range of values used to reference signal nodes when using the **"signalNode":** or **"assignSignal":** keys. This will depend on the product the driver is being used with, as follows:

9.0.0 Netvio IP-JP4-CL-10 Signal Node References

"signalNode": "inN-videotx1" Video signal output node (on input endpoint N)
"signalNode": "inN-audiotx1" Audio signal output node (on input endpoint N)
"signalNode": "inN-infraredtx1" Infrared signal output node (on input endpoint N)
"signalNode": "inN-serialtx1" Serial signal output node (on input endpoint N)
"signalNode": "inN-usbtx1" USB signal output node (on input endpoint N)
"signalNode": "inN-infraredrx1" Infrared signal input node (on input endpoint N)
"signalNode": "inN-serialrx1" Serial signal input node (on input endpoint N)

"signalNode": "outN-videorx1" Video signal input node (on output endpoint N)
"signalNode": "outN-audiorx1" Audio signal input node (on output endpoint N)
"signalNode": "outN-infraredrx1" Infrared signal input node (on output endpoint N)
"signalNode": "outN-serialrx1" Serial signal input node (on output endpoint N)
"signalNode": "outN-usbrx1" USB signal input node (on output endpoint N)
"signalNode": "outN-infraredtx1" Infrared signal output node (on output endpoint N)
"signalNode": "outN-serialtx1" Serial signal output node (on output endpoint N)

Where 1 ≤ N ≤ 128

9.1.0 Netvio MX-H2-0404-10 Signal Node References

All signals cannot be referenced discretely when sending **set_matrix_assign** commands. This product cannot accept **"signalNode":** or **"assignSignal":** keys within **set_matrix assign** commands. Using **get_** commands and any returns and notifications from the API server may include these keys and key values.

"signalNode": "inN-videotx1" Video signal output node (on input endpoint N)
"signalNode": "inN-audiotx1" Audio signal output node (on input endpoint N)

"signalNode": "outN-videorx1" Video signal input node (on output endpoint N)
"signalNode": "outN-audiorx1" Audio signal input node (on output endpoint N)

Where 1 ≤ N ≤ 4

9.2.0 Netvio MX-HT2-0404-10 Signal Node References

Infrared signals cannot be referenced discretely when sending **set_matrix_assign** commands. This product cannot accept **"inN-infraredtx1"** or **"inN-infraredrx1"** values for **"signalNode":** or **"assignSignal":** keys within **set_matrix_assign** commands. Using **get_** commands and any returns and notifications from the API server may include these key values.

"signalNode": "inN-videotx1"	Video signal output node (on input endpoint N)
"signalNode": "inN-audiotx1"	Embedded audio signal output node (on input endpoint N)
"signalNode": "inX-audiotx2"	L/R audio signal output node (on input endpoint X)
"signalNode": "inY-audiotx2"	S/PDIF audio signal output node (on input endpoint Y)
"signalNode": "inN-infraredrx1"	Infrared signal input node (on input endpoint N)

"signalNode": "outN-videorx1"	Video signal input node (on output endpoint N)
"signalNode": "outN-audiorx1"	Embedded audio signal input node (on output endpoint N)
"signalNode": "outN-audiorx2"	L/R & CX audio signal input node (on output endpoint N)
"signalNode": "outN-audiotx2"	HDMI ARC audio signal output node (on output endpoint N)
"signalNode": "outN-audiotx1"	HDbT return audio signal output node (on output endpoint N)
"signalNode": "outN-infraredtx1"	Infrared signal output node (on output endpoint N)

Where 1 ≤ N ≤ 4

Where 1 ≤ X ≤ 2

Where 3 ≤ Y ≤ 4

9.3.0 Netvio MX-HT2-0806-10 Signal Node References

All signals cannot be referenced discretely when sending **set_matrix_assign** commands. This product cannot accept **"signalNode":** or **"assignSignal":** keys within **set_matrix_assign** commands. Using **get_** commands and any returns and notifications from the API server may include these keys and key values.

"signalNode": "inN-videotx1"	Video signal output node (on input endpoint N)
"signalNode": "inN-audiotx1"	Audio signal output node (on input endpoint N)
"signalNode": "inN-infraredrx1"	Infrared signal input node (on input endpoint N)

"signalNode": "outN-videorx1"	Video signal input node (on output endpoint N)
"signalNode": "outN-audiorx1"	Audio signal input node (on output endpoint N)
"signalNode": "outX-infraredtx1"	Infrared signal output node (on output endpoint X)

Where 1 ≤ N ≤ 8

Where 3 ≤ X ≤ 8

9.4.0 Netvio MX-HT2-0808-10 Signal Node References

Infrared signals cannot be referenced discretely when sending **set_matrix_assign** commands. This product cannot accept **"inN-infraredtx1"** or **"inN-infraredrx1"** values for **"signalNode":** or **"assignSignal":** keys within **set_matrix_assign** commands. Using **get_** commands and any returns and notifications from the API server may include these key values.

"signalNode": "inN-videotx1"	Video signal output node (on input endpoint N)
"signalNode": "inN-audiotx1"	Embedded audio signal output node (on input endpoint N)
"signalNode": "inX-audiotx2"	L/R audio signal output node (on input endpoint X)
"signalNode": "inY-audiotx2"	S/PDIF audio signal output node (on input endpoint Y)
"signalNode": "inN-infraredrx1"	Infrared signal input node (on input endpoint N)

"signalNode": "outN-videorx1"	Video signal input node (on output endpoint N)

`"signalNode": "outN-audiorx1"`	Embedded audio signal input node (on output endpoint N)
`"signalNode": "outN-audiorx2"`	L/R & CX audio signal input node (on output endpoint N)
`"signalNode": "outN-audiotx1"`	HDMI ARC audio signal output node (on output endpoint N)
`"signalNode": "outN-audiotx2"`	HDbT return audio signal output node (on output endpoint N)
`"signalNode": "outN-infraredtx1"`	Infrared signal output node (on output endpoint N)

Where 1 ≤ N ≤ 8

Where 1 ≤ X ≤ 2

Where 3 ≤ Y ≤ 4

10 Variables

The following variables will require defining by the Driver user in order to use the Driver:

Product Part Number — The part number of the product.
IP-JP4-CL-10, MX-H2-0404-10, MX-HT2-0404-10, MX-HT2-0806-10, MX-HT2-0808-10
No default.

Controller IP Address — The IPv4 address of an AV over IP Controller or the internal controller within a Matrix Switcher. No default.

Controller TCP Port — The TCP port for the API connection with the AV over IP Controler or the internal controller within a Matrix Switcher.
Default is **443** when the Driver uses WSS.
Default is **23** when the Driver uses raw TCP socket.

Controller Username — The username for the API session with the AV over IP Controler or the internal controller within a Matrix Switcher.
Default is **admin**.

Controller Password — The password for the API session with the AV over IP Controler or the internal controller within a Matrix Switcher.
No default.