NDI Device HTTP API (Version 1.0)

The NDI Device HTTP API allows you to manage and control embedded encoder and decoder devices by sending HTTP requests to specific URLs.

The current HTTP API version is 1.0.

It should be noted that the current HTTP API is not strictly a RESTful API, and it does not follow the formal definition rules of the RESTful API for we believe that the formal definition of RESTful does not do any goods as for device management and control. In any case, the current version of the HTTP API is simple, easy to understand and easy to use. In addition, if you think RESTful API is necessary for you, please feel free to let us know.

> **Compared with RESTful API, NDI device HTTP API (Version 1.0) has the following differences:**
>
> 1. We only use HTTP **GET/POST** requests, rather than PUT/DELETE or other requests;
> 2. We only use JSON format as the return result of each API request (RESTful API also in most cases);
> 3. The result of each API request (success or failure identification), we use the return JSON field to identify it, rather than the HTTP response like RESTful API.

To fully support the HTTP API described in this document, you need to upgrade the firmware of the NDI device to a version with a major version number of 1.50 or higher. Earlier versions may not fully support these APIs or have compatibility issues.

Index

1. Basic Rules

1.1 URL rules

*For **HTTP** access:*

```
http://<host-ip>[:80]/api/v1/<module-name>/<method-name>
```

*For **HTTPS** access:*

```
https://<host-ip>[:443]/api/v1/<module-name>/<method-name>
```

Embedded NDI Devices both support HTTP and HTTPS access, which can be distinguished by URL **http://** or **https://**. The HTTP service port is **80** by default, and the HTTPS service port is **443** by default. `<host-ip>` is the host IP address corresponding to the NDI device you want to request, which is **required**.

In the URL path, **/api/v1** is used to distinguish the version of the HTTP API. This address may change with subsequent version updates.

`<module-name>` is the function module name corresponding to the HTTP API. This document will describe the details of each module in detail; `<module-name>` may be a single word, such as **user**, **tally**, ... etc.; It may also be in the form of paths separated by **/**, such as `decoder/preset`, `decoder/discovery`, ... and so on. See the specific description in the document for details.

`<method-name>` is a method in the corresponding module, for example: /api/v1/user/`add`, **add** is the method of adding users in the user module.

1.2 HTTP Request

You can submit a request to the corresponding HTTP API methods through HTTP **GET** or HTTP **POST**. For methods that require parameters, you can submit parameters in three ways:

A) Submit URL encoded parameters via **HTTP GET** method

E.g:

```
http://<host-ip>/api/v1/tally/set?pgm=1&pvw=0
```

Another example of **url encode special characters** (url encode the spaces in the string "My device"):

```
http://<host-ip>/api/v1/encoder/ndi/set_config?device_name=My%20device
```

B) Submit **application/x-www-form-urlencoded** parameters via **HTTP POST** method

This is the conventional method of submitting parameters by HTTP POST. This means that the Content-Type in the HTTP Request is specified as **application/x-www-form-urlencoded** (HTTP default).

E.g:

```
POST /api/v1/tally/set HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=utf-8
...

pgm=1&pvw=0
```

Another example of **url encode special characters** (url encode spaces in the string "My device"):

```
POST /api/v1/encoder/ndi/set_config HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=utf-8
...

device_name=My%20device
```

C) Submit an object in JSON format via **HTTP POST**; each field in the Object corresponds to a parameter

This requires that the Content-Type in the HTTP request you submit is **application/json** , and the Body of the HTTP request is a valid JSON object.

E.g:

```
POST /api/v1/tally/set HTTP/1.1
Content-Type: application/json;charset=utf-8
...

{"pgm":1, "pvw":0}
```

Another example:

```
POST /api/v1/encoder/ndi/set_config HTTP/1.1
Content-Type: application/json;charset=utf-8
...

{ "device_name": "My device" }
```

If there is no special statement in this document, then the above three methods of submitting HTTP request parameters are all valid and have the same effect.

> You can use tools like **curl, Postman** or write your own simple javascript program to complete the HTTP API test.

1.3 HTTP response and error handling

Unless the Web server of the device has an exception while working, the device will always give you an HTTP **200 OK** response, and include aJSON object in the HTTP Content describing the result of the corresponding API execution.

If the device gives you a response other than 200 OK, it means that the Web server has an abnormal operation. For example, a **404** error indicates that the corresponding request address cannot be found (URL error). Please refer to the HTTP error code and further check the HTTP response error message for the detailed error reason.

Please note: If your HTTP API request address is correct, even if this API fails to execute, the device' s Web server will still return **200 OK**. You should determine the success of the execution by checking the fields in the returned JSON object. This is the biggest difference between the current version of the HTTP API and the RESTful API in terms of execution rules.

If the HTTP API is executed successfully, the returned JSON object format is as follows:

```
{
    "result": "ok",
    "msg": "description message",
    "data": {
        ...
    }
}
```

Among them, **"result"** is a field that must exist, and its value **"ok"** indicates that the requested HTTP API was successfully executed. For successful execution, **"msg"** does not necessarily exist, you can completely ignore its existence, it is only used to describe some additional information. For HTTP APIs that return results, the **"data"** field will exist, and it is a JSON **object** or **array**, which means the API returns the result. This will depend on the specific API, please refer to the specific API description.

If the HTTP API fails to execute, the returned JSON object format is as follows:

```
{
    "result": "error",
    "msg": "error message"
}
```

"result" is a field that must exist. In the current version of the HTTP API, only **"error"** is defined to indicate that the execution failed, and **"auth-failed"** indicates that the security verification failed; but In future versions, it may have other **not "ok"** symbols to represent more error meanings. If "result" is not "ok", you can further check **"msg"** to determine the cause of the error. See the description of each API for details.

Content-Type of HTTP response must be **application/json**.

> **Please note:** In the subsequent API descriptions of this document, if necessary explanations for non-special circumstances, explanations about API request errors will be ignored.

1.4 Charset

The current version of the HTTP API only supports the **utf-8** character set.

1.5 Timeout

When you handle the timeout setting of HTTP requests / responses, please note:

- For most APIs, from accepting the request to completing the execution, the device will complete the response within **0.1s**; but you need to consider the transmission delay and other factors according to the actual network environment, and set a reasonable timeout value (in general It is recommended to set a timeout of 5 seconds or higher).
- However, you need to pay special attention to the following APIs. Their execution / response time should be specially set, and even require special processing (please refer to the corresponding API description for details):
  - /api/v1/sys/reconnect (reset NDI connection)
  - /api/v1/sys/reboot (reboot the device)
  - /api/v1/sys/restore (restore factory settings)
  - /api/v1/mode/switch (switch between encoding and decoding mode, for Connect Spark IO devices)
  - /api/v1/network/set (modify network settings)

1.6 Cross-domain Request

NDI Devices supports HTTP Cross-domain Request, but you need to follow the security rules described in Section 2 of the document.

2. Security Rules

To ensure network security, the HTTP API requires you to perform security authorization according to the security rules described in this section, otherwise the device will reject your HTTP request.

> Unless otherwise specified, we strongly recommend that you use **HTTPS** (not HTTP) to execute the HTTP API. HTTP may bring you hidden security risks of leaking sensitive information (such as user name and password).

2.1 Authorization

Before you use any of the HTTP APIs described in this document, you must first obtain authorization to use the HTTP API.

The mechanism of NDI Devices authorization is briefly described as follows:

1) First of all, you need to provide a valid username and password. NDI Device needs to verify the validity of your username and password;

2) If your username and password are verified correctly, NDI Device will generate a random pair of **Session ID** and **Authorization Token** for you and return it to you in the response.

3) You must record the pair of Session ID and Authorization Token. In each of your next HTTP API requests, you must pass the Session ID and Authorization Token in the HTTP request headers or parameters. There are three alternative delivery methods:

- Through HTTP **GET / POST parameters**;
- Through the HTTP **Headers fields**;
- Through HTTP **cookies**.

Authorization is also an **HTTP API** itself, but it differs from other APIs in that it does not check and verify the Session ID and Authorization Token, but generates the Session ID for you by checking the username and password you submitted And Authorization Token.

API URL

/api/v1/user/authorize

Request

Method: **GET/POST**

Parameter	Value	Description
username	**[STRING] ,Required**	A valid username to request authorization
password	**[STRING] ,Required**	The password of the authorized user

Response

Format (Example) :

```
{
    "result": "ok",
    "data": {
        "session": "559dee0bd779a894618d6e044c35a3fc",
        "token": "4345cd7b092d762bd4a646a98aa9f8ff"
    }
}
```

Data field description:

Field	Value	Description
session	**[STRING]**	Random Session ID The Session ID and Authorization Token will expire within **10 minutes** after **you do not have any HTTP API operations**.
token	**[STRING]**	Random Authorization Token You should record the value of session & token and pass these two values in other subsequent HTTP API requests.

Some advices:

> 1) **Remind you!** It is best not to use the built-in administrator user "admin" for API authorization, which will have serious security risks! You can create other users in the Web UI of NDI Devices and use these users to perform HTTP API requests.
>
> 2) Use HTTPS instead of HTTP to request authorization, otherwise you risk to leak your username and password!

2.2 Pass Session ID and Authorization Token in HTTP API request

If you request authorization according to the method described in 2.1, you will get the Session ID and Authorization Token generated by NDI Device for you. Next, you request any other **HTTP API**, NDI Device will need to verify that your Session ID and Authorization Token are legal. If you do not provide the Session ID and Authorization Token, or if you provide an illegal value, your HTTP API request will return the following error (example):

```
{
    "result": "auth-failed",
    "msg": "...error message..."
}
```

There are three ways to pass Session ID and Authorization Token:

A. [Recommended method] Pass through HTTP Request Headers

In HTTP Request Headers, add **API-Session** Header field to indicate Session ID, and **API-Token** Header field to indicate Authorization Token. E.g:

```
POST /api/v1/your/api HTTP/1.1
...
API-Session: 559dee0bd779a894618d6e044c35a3fc
API-Token: 4345cd7b092d762bd4a646a98aa9f8ff
...

<BODY>
```

If curl is used for testing, the command is as follows:

```
$ curl -H 'API-Session: 4345cd7b092d762bd4a646a98aa9f8ff' -H 'API-Token:
4345cd7b092d762bd4a646a98aa9f8ff' https://192.168.100.168/api/v1/your/api
```

B. Pass through cookies

Pass the Session ID through the HTTP Cookie field **session** and the Authorization Token through the field **token**. E.g:

```
POST /api/v1/your/api HTTP/1.1
...
Cookie:
session=4345cd7b092d762bd4a646a98aa9f8ff;token=4345cd7b092d762bd4a646a98aa9f8ff
...

<BODY>
```

If curl is used for testing, the command is as follows:

```
$ curl -b
'session=4345cd7b092d762bd4a646a98aa9f8ff;token=4345cd7b092d762bd4a646a98aa9f8ff
' https://192.168.100.168/api/v1/your/api
```

C. Pass through GET/POST parameters [not recommended]

If neither Method A nor Method B is applicable to you, you can also pass the Session ID and Authorization Token through the parameters of GET/POST. But we **donot recommend that** you use this method, because it will lessen the standardization of HTTP API parameters, making the parameters of the HTTP API seem confusing.

When it really necessary, you can pass **api-session** as the Session ID and **api-token** as the Authorization Token in the HTTP GET / POST parameters. E.g:

```
https://<device-ip>/api/v1/your/api?api-
session=4345cd7b092d762bd4a646a98aa9f8ff&api-
token=4345cd7b092d762bd4a646a98aa9f8ff
```

3. Status and switching of encoding and decoding modes

Module name: mode

Basic URL: /api/v1/mode/

Some NDI devices have the capabilities of both Encoder and Decoder (such as Connect Spark IO), you can switch between the two. But Connect Spark Plus / TCMI4KUHD only has the capability of Encoder, in this case, mode switching will not work.

3.1 Get the current encoding/decoding working mode

API URL

/api/v1/mode/get

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "mode": "encoder" /*Or "decoder"*/
    }
}
```

Data field description:

Field	Value	Description
mode	**[STRING]**	**"encoder"** or **"decoder"** "**encoder**" identifies the device is working on Encoder mode, and **"decoder"** identifies the device is working on Decoder mode.

3.2 Switch the encoding/decoding working mode

API URL

/api/v1/mode/switch

Request

Method: **GET/POST**

Parameter	Value	Description
mode	**[STRING]**, **Required**	"**encoder**" or "**decoder**" Specify to switch to encoder or decoder mode.

Response

Example:

```
{
    "result": "ok"
}
```

If the handover fails, see the description of the error message in 1.3.

Note: If some device models do not support encoder / decoder mode switching, this operation will not work and return the corresponding error message.

> Switching the Encoder / Decoder mode takes a long time, so if you need to handle the timeout of HTTP requests / responses in particular, please note that the execution time of the request may take up to 5 seconds. Therefore, you need to set the request timeout of the API in order to avoid unexpected failure of the API request due to too short timeout.
>
> If you need to specifically confirm whether the Encoder / Decoder mode is working properly, even after calling this API to switch back you need to further try to call **/api/v1/mode/status** to confirm the working status of the NDI Device. The reason why it looks so strange here is that the Encoder / Decoder mode switching requires a quick Reboot action inside the device.

3.3 Check if the encoding/decoding mode is ready

After you perform the switch operation of Encoder / Decoder mode, due to a quick Reboot action performed inside the device, if you need to confirm whether the switched mode enters the working state normally, you may need to continue to call this API periodically until it returns The result tells you that it has entered the working mode. Usually, it may take 10~15 seconds from switching to working state.

> Don't worry, your current Session ID and Authorization Token are still valid after switching the encoding / decoding mode and performing fast Reboot inside the device.

API URL

/api/v1/mode/status

Request

Method: **GET/POST**

Parameters: NONE

Response

If the encoding / decoding mode is ready, it will return:

```
{
    "result": "ok",
    "data": {
        "mode": "encoder", /*Or "decoder"*/
        "status": "ready"
    }
}
```

If the mode is not yet ready, an error will be returned:

```
{
    "result": "error",
    "reason": "not-ready",
    "msg": "...error message..."
}
```

Please note: **"reason": "not-ready"** may not exist at any time. Put it simple, if the device is performing an internal Reboot process and the program is temporarily inaccessible, the "reason" field will not exist. However, even during the Reboot process, your HTTP request can still be responded to by the device.

4. NDI Encoder status and settings

Module name: encoder/ndi

Basic URL: /api/v1/encoder/ndi/

Note: APIs of this module is only effective when the device works in **Encoder** mode.

4.1 Get NDI Encoding configuration

API URL

/api/v1/encoder/ndi/get_config

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "device_group": "public",
        "device_name": "Spark IO",
        "channel_name": "1916001002",
        "ndi_connection": "multicast",
        "netprefix": "239.254.0.0",
        "netmask": "255.255.0.0"
    }
}
```

Data field descriptions:

Field	Value	Descriptions
device_group	**[STRING]**	NDI Group Name. If the value is "" (empty string) , it means enable NDI default group, that is **public**.
device_name	**[STRING]**	NDI device name, also represents the host name of NDI Device.
channel_name	**[STRING]**	NDI encoding channel name
ndi_connection	**[STRING]**	**"tcp"** or **"multicast"** As NDI Sender, NDI Receiver suggested Connection mode. According to NDI, currently it allows **tcp** (by default) or UDP **multicast** mode. UDP unicast mode will be provided in the future.
netprefix	**[STRING]**	***(Only meaningful when ndi_connection is "multicast")*** A multicast IP address (224.0.0.0 ~ 239.255.255.255) **netprefix** and **netmask** work in pair. Their relation is similar with IP and netmask, namely, netmask decides Subnet of NDI multicast address. NDI SDK will arbitarily select Multicast address from the Subnet for NDI actual transmission.
netmask	**[STRING]**	***(Only meaningful when ndi_connection is "multicast")*** The netmask for multicast IP address
quality	**[INT]**	Valid range of Value: 75 ~ 150 Due to historical reasons, we may select an unappropriate parameter name, the meaning it represents is refer to bitrate recommended by NDI standard, bitrate in percent that we can control is 75%~150%.

> **NOTE**: The current version of HTTP API may include other not-listed fields when returns to the result. These fields exist for software compatibility, please ignore them. In the future software updating, the fields that not listed in the document may be changed or discarded.

4.2 Set NDI Encoding parameter

API URL

/api/v1/encoder/ndi/set_config

Request

Method: **GET/POST**

Parameter	Value	Descriptions
device_group	**[STRING], Optional**	When you specify any one or more parameters of device_group, device_name, channel_name, it will modify Group, Device name or Channel Name. If no parameter is specified, the previously set value will be retained.
device_name	**[STRING], Optional**	
channel_name	**[STRING], Optional**	
ndi_connection	**[STRING], Optional**	Valid value is **"tcp"** or**"multicast"**, specifying the connection mode recommended for NDI Receiver. When specify as **"multicast"**, you need to specify netprefix and netmask parameter at the same time.
netprefix	**[STRING], Optional**	netprefix and netmask work in pair. Their relation is similar with IP and netmask, namely, netmask decides Subnet of NDI multicast address. NDI SDK will arbitarily select Multicast adress from the Subnet for NDI actual transmission.
netmask	**[STRING], Optional**	If you set net_connection="multicast", but not specify netprefix/netmask, the previously set value of netprefix/netmask will be used if they have been set before; otherwise, NDI Device will generate a Multicast address randomly and select 255.255.0.0 as netmask.
quality	**[INT], Optional**	Due to historical reasons, we may select an unappropriate parameter name, the meaning it represents is refer to bitrate recommended by NDI standard, bitrate in percent that we can control is 75% ~ 150%.

Response

Example:

```
{
    "result": "ok"
}
```

If set succeeded, you will get the message result = "ok"; or refer to error message descriptions in 1.3.

Note:

- If NDI Encoding parameter values changed, all current NDI connections will be disconnected, reconfiguring NDI Sender, and it will restart. This process is very fast, normally most NDI Receivers can reconnect in an instant. But please pay attention that **it will reconnect all NDI connections of current NDI Device.**

- All the parameters listed above are optional. If you don't specify any parameter, it means it will do nothing.

4.3 Get NDI Encoding status and video/audio formats information

API URL

/api/v1/encoder/ndi/status

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "video_signal": "hdmi",
        "resolution": "1920x1080p 59.94Hz",
        "xRes": 1920,
        "yRes": 1080,
        "frame_rate": 59.94,
        "interlaced": false,
        "bitrate": 125000, /*in Kbps*/
        "audio_format": "48000 Hz/Stereo",
        "audio_signal": "embedded",
        "audio_sampling": 48000,
        "audio_channels": 2
    }
}
```

Data fields descriptions:

Field	Value	Descriptions
video_signal	**[STRING]**	**none:** no signal input **hdmi:** signal is HDMI, input format identified. **sdi:** signal is SDI, input format identified. video_signal used to show the input signal status of current NDI encoding. When video_signal is**"none"**, it means there no video input from the video input interface (or video format is unidentified or supported) "hdmi" / "sdi" is subject to the device.
resolution	**[STRING]**	Video resolution name Note: This is a user-friendly resolution name, it may not be suitable for your program. You can get program-friendly resolution information via field **xRes/yRes/frame_rate/interlaced**.
xRes	**[INT]**	xRes/yRes shows the width/height of the current video, in pixel. If xRes/yRes = 0 , it means current resolution is invalid.
yRes	**[INT]**	
frame_rate	**[NUMBER]**	Frame rate in fps. If frame_rate = 0 , it means the frame rate of the current video is invalid.
interlaced	**[BOOLEAN]**	If the current video format is interlaced , value will be **true**; otherwise be **false**.
bitrate	**[INT]**	The bitrate of the current NDI encoding (video), in **Kbps**
audio_format	**[STRING]**	**Audio format** Note: This is a user-friendly audio format name, it may not be suitable for your program. You can get program-friendly audio information via **audio_sampling / audio_channels**.
audio_signal	**[STRING]**	**none:** no audio input **embedded**: it's from HDMI/SDI-embedded digital audio **analog:** It's from analog audio input Note: For different devices, the audio source that embedded represents may be different. For HDMI Encoder, it means audio comes from HDMI-embedded digital audio; for SDI Encoder, it means audio comes from SDI-embedded digital audio. Similarly, analog has different meanings for different devices as well. The analog audio input of some devices is **Line In**; while some devices may be **Microphone In**, please distinguish.
audio_sampling	**[INT]**	Audio sampling of the current audio
audio_channels	**[INT]**	Audio channels of the current audio

4.4 Get audio source and volume

API URL

/api/v1/encoder/ndi/get_audio

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "signal": "embedded",
        "volume": 100
    }
}
```

Data fields description:

Field	Value	Descriptions
signal	**[STRING]**	**none:** no audio input **embedded**: It's from HDMI/SDI- embedded digital audio **analog:** It's from analog audio input Note: For different devices, the audio source that embedded represents may be different. For HDMI Encoder, it means audio comes from HDMI-embedded digital audio; for SDI Encoder, it means audio comes from SDI-embedded digital audio. Similarly, analog has different meanings for different devices as well. The analog audio input of some devices is **Line In**; while some devices may be **Microphone In**, please distinguish.
volume	**[INT]**	The current audio volume, valid value range 0~200. 100 means the original audio gain ; <100 means audio decreasing (percentage); >100 audio increasing (percentage).

4.5 Set signal source and/or volume

API URL

/api/v1/encoder/ndi/set_audio

Request

Method: **GET/POST**

Parameter	Value	Descriptions
signal	**[STRING], Optional**	**embedded**: It's from HDMI/SDI-embedded digital audio **analog:** It's from analog audio input Select current audio input. Please refer to 4.4 for audio signal description. If not specify, then not modify the current audio input.
volume	**[INT], Optional**	Adjust the volume of current audio input. Valid value range: 0 ~ 200.100 means the original audio gain ; <100 means audio decreasing (percentage); >100 audio increasing (percentage). If not specify, then not modify the current audio input.

Response

Example:

```
{
    "result": "ok",
}
```

If set successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

5. NDI Sources discovery

Module name: decoder/discovery

Basic URL: /api/v1/decoder/discovery/

Note: : APIs of this module is only effective when the device works in **Decoder** mode.

5.1 Get NDI Sources discovered in network

API URL

/api/v1/decoder/discovery/get

Request

Method: **GET/POST**

Parameter	Value	Description
force	**[ANY], Optional**	If you specify **force** parameter, no matter what it is, it requires NDI Device to force a rescan of the network.

Response

Example:

```
{
    "result": "ok",
    "data": [
        {
            "group": "public",
            "name": "Spark_IO-1916001002 (Channel 1)",
            "device_name": "Spark_IO-1916001002",
            "channel_name": "Channel 1",
            "url": "192.168.100.168:5961"
        },
        /* ... Each discovered items in the array ... */
    ],
    "data_size": 10
}
```

Data field description:

"data" is aJSON array, "data_size" is a field that help assist to describe "data" array size (you may not use it) .

Each item of the array means one NDI Source discovered in network:

Field	Value	Description
group	**[STRING]**	NDI Source located NDI Group. If the Group you get is a empty string **""**, it means public group by default.
name	**[STRING]**	NDI Source Name. It is original name for NDI Source. If you want to get device name, channel name, please refer to device_name and channel_name.
device_name	**[STRING]**	NDI Device Name resolved from **name**, which normally represent host name of NDI Source.
		NDI Channel Name resolved from **name**
url	**[STRING]**	NDI Source URL. **Note**: Current URL format is `<IP>:<port>`, but it is recommended to not take it as the fact, NDI SDK does not guarantee this. Please keep the url original value and don't attempt to change it while using.

TIPS:

> A. NDI Discovery based on mDNS mechanism, which can only discover NDI Sources that in the same subnetwork with current NDI Device by default.

> B. Considering there a huge lot of NDI Sources in your network, and NDI Discovery speed is not as fast as you imagine, so NDI Device will cache the sources discovered previously and return the results to you as soon as possible. In the same time, NDI Device will keep scanning and update cache in the background. Therefor, the result you get may not be completely up-to-date (but in most cases). You can get updates by calling this API periodically.

5.2 Manual discovery: specify IP and/or NDI Groups

In most cases, we can discover automatically NDI Sources via network, but there some special conditions:

- NDI auto discovery can't discover NDI Sources not in the same Subnet;
- NDI has its own logical group, by default, NDI auto discovery can only discover devices of group **"public"**;
- Other conditions that due to network restricted network **Multicast**.

Under above conditions, we may need to manually specify target IP address and/or NDI Groups of NDI discovery.

API URL

/api/v1/decoder/discovery/set_manual_targets

Request

Method: **POST** (`Please note` This API can only use POST method and submitted parameters need to use JSON format)

Parameters (example):

```
{
    "ip": ["192.168.100.3", "172.16.10.5" /*, ...*/],
    "group_name": ["public", "private_group", "my-group" /*,...*/]
}
```

Parameter	Value	Description
ip	**[ARRAY(STRING)]**, **Optional**	Through JSON array, specifying one or more target IP addresses of NDI Sources that NDI Sources manually discovered. Specified IP address can be in different Subnet with current NDI Device. This parameter is optional. If not specify, the former specified manual IPs will be cleared.
group_name	**[ARRAY(STRING)]**, **Optional**	Group Names. Through JSON array, specifying one or more NDI Group Names that NDI Sources manually discovered This parameter is optional. If not specified, the former specified Group Name will be cleared.

Note:

A) The IP and NDI Groups that you manually specified will be recorded and saved by the device. Therefore, even if you restart the device, these IP and NDI Groups will work.

B) Manually specifying IP and NDI Groups will not influence it auto discovery. Therefore, you will see all the auto-discovery NDI Sources and manually specified Sources (if target exists).

C) As showed in the parameter description, if you do not specify "ip" or "group_name", it means that it will clear the previous settings. If you need to keep the previous settings, be sure to pass the original value as a parameter. You can get the previous value via API **/api/v1/decoder/discovery/get_manual_targets**.

Response

Example:

```
{
    "result": "ok"
}
```

If set successfully, it will return the message of result = "ok". Otherwise, please refer to 1.3 for error message descriptions.

5.3 Get specified manual discovery and NDI Groups

API URL

/api/v1/decoder/discovery/get_manual_targets

Request

Method: **GET/POST**

Parameter: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "ip": ["192.168.100.3", "172.16.10.5" /*, ...*/],
        "group_name": ["public", "private_group", "my-group" /*,...*/]
    }
}
```

Data field description:

Field	Value	Description
ip	**[ARRAY(STRING)]**	Return a list of all manually specified IP addresses through the array.
group_name	**[ARRAY(STRING)]**	Return a list of all manually specified NDI Groups through the array.

6. NDI decoding Preset

Our NDI decoder has a very practical function: **Preset**. Preset means you can add the discovered NDI Source to 9 presets represented by 1~9, which is your "Favorites". When you need to quickly select the NDI Source added in Preset for decoding, you only need to specify the Preset ID (1~9).

Preset can bring some additional functional advantages to our NDI decoding devices. For example, we will support an external physical keyboard in the future. Press the 1~9 on the keyboard to complete the rapid switch.

There is also a special **preset-0** on the NDI decoder. The meaning of this preset is Blank, used to fill the screen with a customizable color.

Module name: decoder/preset

Basic URL: /api/v1/decoder/preset/

6.1 Get current preset list and status

API URL

/api/v1/decoder/preset/status

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:


```
{
    "result": "ok",
    "data": [
        {
            "id": "1",
            "enable": 1, /*or 0*/
            "group": "public",
            "name": "Spark_IO-1916001002 (Channel 1)",
            "device_name": "Spark_IO-1916001002",
            "channel_name": "Channel 1",
```

```
            "url": "192.168.100.168:5961" ,
            "ip": "192.168.100.168",
            "online": "on", /*or "off"*/
            "current": true /*or false*/
        },
        /*...*/
        {
            "id": "0",
            "current": true, /*or false*/
            "color": "#000000"
        }
    ],

    "data_size": 10
}
```

Data field description:

"data" is aJSON array, "data_size" is a field that help assist to describe "data" array size (you may not use it) .

Each item of the array means one NDI Source discovered in network:

Field	Value	Description
id	**[STRING]**	1~9 represents normal NDI Source Preset; 0 is Blank
enable	**[INT]**	1: This Preset is enabled 0: This Preset doesn't enable NDI Source; in this case, all subsequent fields are meaningless.
group	**[STRING]**	Preset saved group name of NDI Source
name	**[STRING]**	Preset saved name of NDI Source
device_name	**[STRING]**	Device Name of NDI Source extracted from the `name` field
channel_name	**[STRING]**	Channel Name of NDI Source extracted from the `name` field
url	**[STRING]**	Original URL of NDI Source
ip	**[STRING]**	IP address of NDI Source extracted from `url` field
online	**[STRING]**	According to NDI discovery results, it detects the online status of NDI Source defined by this preset: **on:** current NDI Source online (It can be discovered) **off:** current NDI Source not online (It can't be discovered) **Note:** online status is decided by NDI discovery results. It doesn't mean whether NDI source can be connected and obtain video data. Therefore, it only provides reference significance and cannot be used as a basis for the device to determine online.
current	**[BOOLEAN]**	**true:** The preset is NDI Source of the current decoding **false:** The preset is not NDI Source of the current decoding
color	**[STRING]**	**Only valid for id = "0"** For Preset 0 (Blank) , it doesn't include the following fields `enable` , `group` , `name` , `device_name` , `channel_name` , `url` , `ip` , `online` , while return color represents Blank fill color. For detailed formats please refer to 6.4 `/api/v1/decoder/preset/set_blank`

6.2 Add NDI Source to a specified preset

API URL

/api/v1/decoder/preset/add

Request

Method: **GET/POST**

Parameter	Value	Description
id	**[STRING], Required**	Specify preset ID , valid range is 1~9
name	**[STRING], Required**	NDI source name , you can get discovery NDI sources list by API `/api/v1/decoder/discovery/get` and extract name field from them.
url	**[STRING], Required**	Original URL of NDI source , you can get discovery NDI sources list by API `/api/v1/decoder/discovery/get` and extract url field from them.
group	**[STRING], Optional**	NDI group name of NDI source. The parameter is optional, If you can get by API `/api/v1/decoder/discovery/get` and extract group field, it is best to pass in as a parameter.

Note: If the preset ID you specified originally has an NDI source definition, the newly added NDI source will overwrite the old definition.

Response

Example:

```
{
    "result": "ok"
}
```

If add successfully, it will return the message of result = "ok". Otherwise, please refer to 1.3 for error message descriptions.

6.3 Remove NDI source definition of preset

API URL

/api/v1/decoder/preset/remove

Request

Method: **GET/POST**

Parameter	Value	Description
id	**[STRING], Required**	Specify the preset ID for removing , valid range 1~9

Response

Example:

```
{
    "result": "ok"
}
```

If remove successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

6.4 Define preset 0 (Blank) color

API URL

/api/v1/decoder/preset/set_blank

Request

Method: **GET/POST**

Parameter	Value	Description
color	**[STRING], Required**	Specify the color of preset 0 (Blank). The description format of colors is consistent with the description of colors in HTML, namely: 1. Use RGBA color 2. Start with #, hex to describe R, G, B and A respectively Common color description types: #RGB (E.g. **#f00** means red) #RRGGBB (E.g. **#ff00ff** means purple) #RRGGBBAA (E.g. **#00ff0080** means green, 50% transparency) **Note**: The current version doesn't support Alpha function.

Response

Example:

```
{
    "result": "ok"
}
```

If set successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

7. Current decoding output

Module name: decoder/current

Basic URL: /api/v1/decoder/current/

7.1 Get current decoding status

API URL

/api/v1/decoder/current/status

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "name": "Spark_IO-1916001002 (Channel 1)",
        "url": "192.168.100.168:5961",
        "ip": "192.168.100.168",
        "online": true, /*or false*/
        "resolution": "1920x1080p 59.94Hz",
        "codec": "SHQ7",
        "xRes": 1920,
        "yRes": 1080,
        "frame_rate": 59.94,
        "inst_frame_rate": 59.93,
        "interlaced": false,
        "de_interlace": false,
        "bitrate": 125000,
        "audio_format": "48KHz / 2CH",
        "audio_sampling": 48000,
        "audio_channels": 2,
        "warning": ""
    }
}
```

Data field description:

Field	Value	Description
name	**[STRING]**	The NDI Source Name that currently decoded. If currently output Preset 0 (Blank), then name should be **"(Blank)"**
url	**[STRING]**	The original URL of NDI Source that currently decoded. If there no decoding NDI Source , then url = "" (empty strings)
ip	**[STRING]**	IP address of of NDI Source that currently decoded. If no decoding, ip = "0.0.0.0"
online	**[BOOLEAN]**	**true**: Current NDI Source is online; **false**: current NDI Source is not online
resolution	**[STRING]**	Resolution of the current NDI Source (friendly name). If you need to obtain the program-friendly information of the video, such as width/height, please refer to `xRes`, `yRes`, `frame_rate`, `interlaced` fields.
codec	**[STRING]**	CODEC ID of of the current NDI Source: **SHQ0 / shq0**: NDI YUV4:2:0 format encoding **SHQ2 / shq2**: NDI YUV4:2:2 format encoding **SHQ7 / shq7**: NDI YUVA4:2:2:4 format encoding Other CODEC ID may exist, but can't be supported yet.
xRes	**[INT]**	Video resolution width of the current NDI Source
yRes	**[INT]**	Video resolution height of the current NDI Source
frame_rate	**[NUMBER]**	Frame-rate of the current NDI Source. **Please note**: `frame_rate` means the original frame_rate of NDI Source. The other field `inst_frame_rate` means (Instant Frame Rate), inst_frame_rate means the actual frame rate calculated by the program.
inst_frame_rate	**[NUMBER]**	Instant Frame Rate , the actual framerate calculated by the program.
interlaced	**[BOOLEAN]**	**true**: NDI Source is Interlaced video **false**: NDI Source is progressive video

Field	Value	Description
de_interlace	**[BOOLEAN]**	The NDI decoder has the ability to execute deinterlace of interlaced video and output it in progressive format. It needs you to configure via API `8.3 /api/v1/decoder/output/set` (deinterlace operation is not performed by default). If you configure deinterlace, and the original video of the current NDI Source is interlaced , then `de_interlace` will be true , which used to instruct you that the Decoder has now undergone deinterlace conversion.
bitrate	**[INT]**	Current real-time bitrate statistics, in **Kbps**
audio_format	**[STRING]**	The current audio formats (friendly name). If you need to get the program-friendly information of the audio, please refer to `audio_sampling` , `audio_channels`
audio_sampling	**[INT]**	Audio sampling of the current audio
audio_channels	**[INT]**	Channels of of the current audio
warning	**[STRING]**	The following defined string ID (**bold text**) is used to describe warnings/errors that occur when decoding output: **WARN:url-changed** Meaning: NDI source URL changed **WARN:offline** Meaning: Offline **ERROR:no-video-output** Meaning: Video output config error **ERROR:no-audio-output** Meaning: Audio output config error **WARN:invalid** Meaning: Invalid output configuration **WARN:match-error** Meaning: Format/resolution is unsupported **WARN:unsupported-codec** Meaning: CODEC is unsupported **WARN:unsupported-resolution** Meaning: Resolution is unsupported: **WARN:url-changed** Meaning: NDI source URL changed

Field	Value	Description
		WARN:offline Meaning: Offline **ERROR:no-video-output** Meaning: Video output config error **ERROR:no-audio-output** Meaning: Audio output config error **WARN:invalid** Meaning: Invalid output configuration **WARN:match-error** Meaning: Format/resolution is unsupported **WARN:unsupported-codec** Meaning: CODEC is unsupported **WARN:unsupported-resolution** Meaning: Resolution is unsupported

7.2 Switch preset or specified NDI source to the current decoding output

API URL

/api/v1/decoder/current/set

Request

Method: **GET/POST**

Parameter	Value	Description
id	**[STRING], Optional**	Specify preset ID (0~9), switch the NDI source saved in preset ID to the current decoding output. This parameter is optional. If ID is specified, the following other parameters will be meaningless; if ID is not specified, you must specify the following `name` and `url` parameters. If you specify **id = "0"**, it means that Blank is output directly without decoding. Please refer to section `6` for preset 0 description.
name	**[STRING], Optional**	If you don't specify the parameter of the above `id`, then `name` and `url` parameter is a must; otherwise no need to specify. name is the name of NDI Source, usually you can get via 5.1 `/api/v1/decoder/discovery/get`. According to the current rules of NDI SDK, `url` is the key parameter to get NDI Source connection, name only providing reference for discovering NDI source. So, in a sense, you can ignore name or provide an arbitrary string. **But we strongly recommend** that you should provide the correct and legal name.
url	**[STRING], Optional**	The original URL of NDI source. If you don't specify parameter `id`, then url parameter is required.

Response

Example:

```
{
    "result": "ok"
}
```

If set successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

8. Advanced options of encoding output and switching

We believe that you expect a smooth and even switching with no video freezing, screen flicker or other bad phenomenon, when switching different NDI source to decode outputting.

But we need to tell you something from a technical perspective:

- It will take some time from building an NDI source connection to get actual video/audio data. Maybe it's 0.1 second, 0.2 second and more. (depend on your network status)
- The resolution and frame rate will to be changed if switching to new NDI source, this makes us have to reconfigure the output format of HDMI/SDI. It is terrible, the screen probably

appears no outputting or flicking shortly.

All above will bring us bad experiences.

We will try our best to help you solve these problems. In possible cases, we will provide some settable options to solve image freezing and flicker issues. Of course, some of the issues can not to be solved due to the limitation of NDI decoder hardware capability.

NDI decoder can solve image freezing and flicker well in the following situation:

- By setting smooth switch and timeout waiting through `8.1 /api/v1/decoder/output/set_smoothness`, the stagnantly of NDI connection wait time can be fixed.
- `8.3 /api/v1/decoder/output/set` forced frame rate outputting can solve it in the situation of resolution unchanged but frame rate changed.
- By using `8.3 /api/v1/decoder/output/set` command to output the format of deinterlace, it can convert interlaced video into progressive output, to prevent display issues caused by video changes of interlaced/progressive.

In the following cases, it will probably occur flicker issue due to hardware capability limitation.

- Resolution changed

Module name: decoder/output

Basic URL: /api/v1/decoder/output/

8.1 Time-out of smooth switch setting

In order to solve the issue of time-out on NDI connecting, we adopted such strategy:

When you initiate a new NDI source switch request, NDI decoder will keep current decode running and trying to connect to a new assigned NDI source. If the new source is connected successful and the video/audio data is received, NDI decoder will immediately stop decoding old NDI source and switch to new NDI source. Then, the switching is "as smooth as silk".

Well, the real situation is always unsatisfactory. There are several cases:

- You network is terrible. New NDI source connecting has taken a long time to work successful. If NDI decoder continuously output the old NDI source by following above methods, it gives you a illusion that the initiated NDI source switching is failed.
- New NDI source is not reachable.

In order to deal with any of these situations, our strategies is: you can specify a time of timeout with a NDI source connecting, if source connection is completed within this time, well, that's what you expected' and if source can not to be connected in time, unfortunately, we must force to stop decode outputting and wait for new source until it is ready.

If you set up timeout as 0, you might guess what happened——it will stop old NDI source immediately and connect to new source. It will cause image stopping.

API URL

/api/v1/decoder/output/set_smoothness

Request

Method: **GET/POST**

Parameter	Value	Description
timeout	**[INT], Required**	Specify timeout of smooth switching. **in ms.** if it is 0, no smooth switch happen.

Response

Example:

```
{
    "result": "ok"
}
```

If set successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

8.2 Get timeout of smooth switching

API URL

/api/v1/decoder/output/get_smoothness

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

```
{
    "result": "ok",
    "data": {
        "timeout": 200
    }
}
```

Data field description:

Field	Value	Description
timeout	**[INT]**	Timeout of current smooth switching , in **ms**.

8.3 Advanced options of video/audio output

The default rule of video/audio output is by following the format of NDI source, that' s the same resolution/frame rate/audio sampling rate/channels as NDI source.

But you can specify the advanced option of resolution, frame rate and audio sampling rate.

API URL

/api/v1/decoder/output/set

Request

Method: **GET/POST**

Parameter	Value	Description
resolution	**[STRING], Optional**	You can set up the following option value: **auto**: default , follow NDI source resolution **deint**: If the video format of NDI source is interlaced , it will deinterlace to progressive format automatically (1080i -> 1080p conversion). If the Source format is progressive, no affected.
frame_rate	**[INT], Optional**	Specify frame rate output. optional value include: **23.98, 24, 25, 29.97, 30, 50, 59.94, 60** and one specified value: **0** 0 mean use the same frame value as NDI Source. if specified an effective frame rate value , NDI decoder will force output the specified frame rate. For example , the original video format of NDI source is 1920x1080p 60Hz , but if specify frame_rate as 29.97 , NDI decoder will force output as 1920x1080p 29.97Hz.
sample_rate	**[INT], Optional**	Specify audio sampling rate. **pay attention:** due to the version of hardware limitation , sample_rate can only be set up 48000. So , please miss the this parameter.

Response

Example:

```
{
    "result": "ok"
}
```

If set successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

8.4 Get video/audio advanced output options

API URL

/api/v1/decoder/output/get

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

```
{
    "result": "ok",
    "data": {
        "resolution": "auto",
        "frame_rate": 59.94,
        "sample_rate": 48000
    }
}
```

Field	Value	Description
resolution	**[STRING]**	Refer to 8.3 `/api/v1/decoder/output/set` description
frame_rate	**[INT]**	Refer to 8.3 `/api/v1/decoder/output/set` description
sample_rate	**[INT]**	Refer to 8.3 `/api/v1/decoder/output/set` description

9. Tally status and control

Module name: tally

Basic URL: /api/v1/tally/

Tally status and control act both on Encoder and Decoder mode. Under Encoder mode , you can get current Tally status through API, even control Tally light's on/off (although it looks like it is not necessary); under Decoder mode , you can set up Tally status (Program On/Off, Preview On/Off) will send to the decoding NDI source.

9.1 Get current Tally status

API URL

/api/v1/tally/status

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "pgm": 1,
        "pvw": 0
    }
}
```

Data field description:

Field	Value	Description
pgm	**[INT]**	1: PGM(Program) On 0: PGM(Program) Off
pvw	**[INT]**	1: PVW(Preview) On 0: PVW(Preview) Off

9.2 Set current Tally status

API URL

/api/v1/tally/set

Request

Method: **GET/POST**

Parameter	Value	Description
pgm	**[INT], Optional**	1: Set Program On 0: Set Program Off not specified: keep previous status
pvw	**[INT], Optional**	1: 设置Preview On 0: Set Preview Off not specified: keep previous status

Response

Example:

```
{
    "result": "ok"
}
```

If set successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

Note:

- If current NDI Device is in Encoder mode and it is connected by one of NDI Receiver, as NDI Receiver can probably update Tally status, so setup of Tally may be covered by updated Tally status. Please pay attention to this detail.
- If NDI is in Decoder mode, the Tally status which set will send to the NDI source which is decoding; if you switch to another NDI source, it will clear the previous Tally status of NDI source, and send new status to new NDI Source.

10. System management and control

Module name: sys

Basic URL: /api/v1/sys/

10.1 Get working status of device

API URL

/api/v1/sys/server_info

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:


```
{
    "result": "ok",
    "data": {
        "addr": "192.168.0.100",
        "port": 443,
        "name": "https://192.168.0.100",
        "persis": "4H 15M 32S",
        "start_time": "2020-05-03 12:10:09",
        "cpu_cores": 2,
        "cpu_payload": 33,
        "mem_used": 124.06,
        "mem_total": 620.00
    }
}
```

Field	Value	Description
addr	**[STRING]**	IP address of current device
port	**[INT]**	Current visit access port of HTTP API requests (default HTTP: 80, HTTPS: 443)
name	**[STRING]**	**Note:** This is a field to be discarded or changed. Current it means the host of you are visiting HTTP API.
persis	**[STRING]**	The persistent time of device running, format: `<hour>` H `<minute>` M `<second>` S
start_time	**[STRING]**	start-up time of device , format: Year-Month-Day Hour:Minute:Second
cpu_cores	**[INT]**	CPU cores of the current device
cpu_payload	**[INT]**	Current CPU load %
mem_used	**[NUM]**	Current memory in-using, **MB**
mem_total	**[NUM]**	Total memory size, **MB**

10.2 Reset all NDI connections

This will disconnect all NDI connections and restart: for NDI Encoder, it means reinitialize NDI Sender; for NDI Decoder, it will disconnect current NDI Source and re-connect.

reset time is about 4-6 seconds, please attention.

API URL

/api/v1/sys/reconnect

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

```
{
    "result": "ok"
}
```

If execute successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

10.3 Reboot device

This operation will control device reboot.

call this API, HTTP request will return immediately, but the real reboot action of device will start in **3s** after API executed successfully, the whole reboot process last about 20s. Because the network of device usually use DHCP to obtain address, and the actual DHCP obtain time is depend on the actual network status, so, the expected device visit time will >= 30s.

You will not allowed to request any HTTP API until the device finished rebooting.

Please pay attention: Your security credential (Session ID and Authorization Token) will be **invalidated** after rebooting, you must re-authorize HTTP API by using the method which shown in `chapter 2`.

API URL

/api/v1/sys/reboot

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

```
{
    "result": "ok"
}
```

If execute successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

10.4 Restore factory setting

This operation will restore NDI device to factory default setting. The affected setting include:

- The method to get the network address will be restored to DHCP;
- The network's Failsafe address will be restored to the factory default (192.168.100.168, or 192.168.1.168, depending on different devices, please refer to the product instruction manual) ;
- NDI group will be restored to default (public);
- NDI device name will be restored to the format of **"<PRODUCT_TYPE>-<SERIAL-NUMBER>"** ;
- NDI channel name will be restored to the factory default name (Usually **Channel-1**, but please subject to actual product) ;
- The type of NDI connection will be restored to default **(TCP)**;
- NDI decoder's current decoding source and preset will be cleared, the color of Preset 0 (Blank) will to be restored to default **(black)** ;
- All users will to be cleared, **admin** user will restore to the default password.

Call this API, HTTP will return immediately, the real reboot action of device will start in 3s after API executed successfully, the reboot action and impact are the same as described in `10.3`

API URL

/api/v1/sys/restore

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

```
{
    "result": "ok"
}
```

If execute successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

11. Network Configuration

Module name: network

Basic URL: /api/v1/network/

11.1 Get current network configuration

API URL

/api/v1/network/get

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

```
{
    "result": "ok",
    "data": [
      {
        "device": "eth0",
        "state": "up",
```

```
            "ip": "192.168.2.160",
            "mask": "255.255.255.0",
            "mac": "68:3A:7F:8C:A7:96",
            "gw": "192.168.2.1",
            "dynamic": "y",
            "dns": "8.8.8.8; 4.4.4.4",

            "ip2": "192.168.100.168",
            "mask2": "255.255.255.0",
            "gw2": "",

            "ip3": "0.0.0.0",
            "mask3": "255.255.255.0",
            "gw3": ""
          }
        ],
        "data_size": 1
}
```


Data field description:

"data" is aJSON array, "data_size" is a field that help assist to describe "data" array size (you may not use it) .

Each item of the array means one network interface's configuration:

Field	Value	Description
device	**[STRING]**	Device name of current network interface. **pay attention** this device name, it is the basis for you to modify the network configuration. it is said that when you modify the network configuration, you must specify the device name.
state	**[STRING]**	Current working status of the network: **up**: The network is working fine **down**: Disconnected because the network cable disconnected **disabled**: Forbidden to use the network for other reasons **error**: Network work failure
ip	**[STRING]**	Current IP address of network configuration. **Note**: If your network is configured with DHCP address, ip will show the current IP address obtained by DHCP ; and if the network is configured with a static IP address, ip will shows the static IP address that you set.
mask	**[STRING]**	Subnet mask of the current network (netmask)
mac	**[STRING]**	MAC address of current network card
gw	**[STRING]**	The default gateway of the current network (gateway). If it is not set yet, it may be "" (empty string) or "0.0.0.0"
dynamic	**[STRING]**	**y**: Start DHCP configuration **n**: Use static IP address configuration
dns	**[STRING]**	current configured DNS server address. Separate multiple addresses with ';'.
ip2	**[STRING]**	The second IP address of the device (also called Failsafe address). Because the modification of the main IP address of the device may occur error or cannot access the device due to forgotten, the Failsafe address allows you to configure a default, easy-to-remember IP. So, you can access and configure the device through the Failsafe address. So, we suggest you not to modify ip2 (Failsafe address) as much as possible.
mask2	**[STRING]**	Subnet mask of ip2 (netmask)
gw2	**[STRING]**	The default gateway of ip2 (gateway). If not set, it may be "" (empty string) or "0.0.0.0"
ip3/mask3/gw3	**[STRING]**	**ip3/mask3/gw3** and **ip2/mask2/gw2** have the similar meanings and uses. But you do not need to configure ip3 in most cases, **We do not guarantee the validity of this parameter configuration.**

11.2 Modify network configuration

Important note: The modification of the network configuration is a sensitive operation, the wrong configuration may lead to the device can not be normal access. Therefore, we remind you:

- Regular network address modification, please try to finish through the Web UI of NDI device;
- If it is unable to access due to network parameter is modified, you can restore to factory default settings by pressing the **Reset button** of the device.

API URL

/api/v1/network/set

Request

Method: **GET/POST**

Field	Value	Description
device	**[STRING], Required**	Specify the name of the network interface to modify. Please refer to 11.1 `/api/v1/network/get` description of the device parameter. This parameter is necessary.
dynamic	**[STRING], Optional**	**y**: use DHCP to get IP address. If dynamic="y", then `ip`, `mask`, `gw`, `dns` will make no sense **n**: use static IP address
ip	**[STRING], Optional**	configure IP address
mask	**[STRING], Optional**	subnet mask (netmask)
mac	**[STRING], Optional**	if mac specified, MAC address of the current network card can be modified. Please pay attention: 1. You must provide a valid and effective MAC address; 2. Unless you are sure what you want to do, or please do not modify MAC address.
gw	**[STRING], Optional**	default gateway(gateway). specify "" (empty string) means no gateway configuration.
dns	**[STRING], Optional**	specify 0 or more DNS server address. Separate multiple addresses with ';'. "" (empty string) means no configuration.
ip2	**[STRING], Optional**	configure Failsafe IP address. If you need to clear Failsafe address, please setup ip2 = "" (empty string)
mask2	**[STRING], Optional**	subnet mask of Failsafe address (netmask)
gw2	**[STRING], Optional**	default gateway which correspond to Failsafe address (gateway). specify "" (empty string) means no gateway configuration.
ip3/mask3/gw3	**[STRING], Optional**	ip3/mask3/gw3 and **ip2/mask2/gw2** have similar meanings and uses. But in most cases, you do not need configure ip3. **We do not guarantee the validity of this parameter configuration**

Response

Example:

```
{
    "result": "ok"
}
```

If execute successfully, it will return the message of result = "ok", or please refer to 1.3 for error message descriptions.

Note: You must use new IP address to access HTTP API if network address was modified and IP address was changed. The effectiveness of network address modification is usually <1s. If you specified IP address for DHCP acquisition, it depends on your actual network environment, the time to obtain a valid IP address is uncertain.