KILOVIEW

领先的视频 IP 化传输解决方案专家

API 说明文档

N5/N6 视频编解码器

NDI High Bandwidth+NDI|HX 全能编解码器

(2023-8 版)

通过本手册，您可以了解到

- API 的接口说明
- API 的使用方法

由于产品不断升级更新，您所购买的产品可能与本手册的内容有所变更，请以包装箱内产品实际为准。

法律声明

若接收长沙千视电子科技有限公司（以下称为“千视电子”）的此份文档，即表示您已同意以下条款。若不同意以下条款，请停止使用本文档。

本文档版权所有长沙千视电子科技有限公司。保留任何未在本文档中明示授予的权利。文档中涉及千视电子的专有信息。未经千视电子事先书面许可，任何单位和个人不得复制、传递、分发、使用和泄漏该文档以及该文档包含的任何图片、表格、数据及其他信息。

KILOVIEW 千视是千视电子的注册商标。千视电子产品的名称和标志是千视电子的商标或注册商标。在本文档中提及的其他产品或公司名称可能是其各自所有者的商标或注册商标。在未经千视电子或第三方权利人事先书面同意的情况下，阅读本文档并不表示以默示、不可反言或其他方式授予阅读者任何使用本文档中出现的任何标记的权利。

本产品符合有关环境保护和人身安全方面的设计要求，产品的存放、使用和弃置应遵照产品手册、相关合同或相关国法律、法规的要求进行。

本文档按“现状”和“仅此状态”提供。本文档中的信息随着千视电子产品和技术的进步将不断更新，千视电子不再通知此类信息的更新。

本文档未尽事宜，请访问千视电子网站 www.kiloview.com 获取相关信息和技术支持。

目录

1. 安全性规则

为保证网络安全性，HTTP API 需要您按照本节所描述的安全性规则进行安全授权，否则设备将拒绝您的 HTTP 请求。

如非特殊情况，我们强烈建议您使用 **HTTPS**（而不是 HTTP）来执行 HTTP API。HTTP 可能为您带来泄露敏感信息（例如用户名、密码）的安全隐患。

1.1 授权

在您使用任何本文档所描述的 HTTP API 之前，您必须先获得使用 HTTP API 的授权。

NDI Devices 授权的机制简单描述如下：

1) 首先，您需要提供一个有效的用户名和密码，NDI Device 需要校验您的用户名和密码的合法性；

2) 如果您的用户名和密码校验正确，NDI Device 将为您生成一对随机的 **Authorization Token**，并在响应中返回给您。

3) 您必须记录 Authorization Token。在您接下来的每一个 HTTP API 请求中，您都必须传递 Authorization Token 在 HTTP 的请求 Cookie 中。

授权本身也是一个 HTTP API，不过它与其它的 API 不同之处在于，它不检查和校验 Authorization Token，而是通过检查您提交的用户名和密码来为您生成 Authorization Token。

API URL

/api/user/authorize.json

Request

Method：**GET/POST**

参数	Value	说明
user	[STRING] ,Required	请求授权的有效用户名。
password	[STRING] ,Required	请求授权用户的密码。

Response

格式（**Example**）：

```
{
   "result": "ok",
   "data": {
       "token": "2e33a9b0b06aa0a01ede70995674ee23",
       "alias": "Admin"
  }
}
```

Data 字段说明：

Field	Value	说明
token	[STRING]	随机的 Authorization Token。
alisa	[STRING]	用户名称别名

一些建议：

1) 提醒您注意！最好不要使用系统内建的管理员用户"admin"进行 API 授权，这会有严重的安全风险！您可以在 NDI Devices 的 Web UI 中创建其他的用户，并使用这些用户来执行 HTTP API 请求。

2) 使用 HTTPS 而不是 HTTP 来请求授权，否则您有泄露用户名和密码的风险！

1.2 在 HTTP API 请求中传递 Authorization Token

如果您按照 1.1 所描述的方法请求授权成功，您将获得 NDI Device 为您生成 Authorization Token。接下来，您**请求任何其它的 HTTP API**，NDI Device 都将需要验证您的 Authorization Token 是否合法。如果您没有提供 Authorization Token，或者您提供了非法的值，您的 HTTP API 请求将返回如下错误 (example)：

```
{
   "result": "auth-failed",
   "msg": "Invalid token"
}
```

通过 Cookie 传递

通过 HTTP Cookie 字段 **" token "** 传递 Authorization Token。例如:

```
POST /api/your/api.json HTTP/1.1
...
Cookie: token=9b6bdc43a8884cbd9503a1cda00a2d3b
...
```

```
<BODY>
```

如果使用 curl 进行测试，命令如下：

```
$ curl -b 'token=9b6bdc43a8884cbd9503a1cda00a2d3b'
https://192.168.100.168/api/your/api.json
```

1.3 取消身份验证(不建议)

可以在网页端的用户管理界面关闭 HTTP API Authorization ,这样调用 api 将不再进行用户验证

2.编码和解码模式的状态及切换

Module name: mode

Basic URL: /api/mode/

N6 同时具备 Encoder 和 Decoder 的能力，您可以在这两者之间进行切换。

2.1 获取当前编码/解码工作模式

API URL

/api/mode/get.json

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

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

Data 字段说明：

Field	Value	说明<
mode	[STRING]	"encoder" 或 "decoder" 返回 "encoder"表示设备工作在 Encoder 模式；"decoder"表示设备工作在 Decoder 模式

2.2 切换编码/解码工作模式

API URL

/api/mode/switch.json

Request

Method: **GET/POST**

参数	Value	说明
mode	[STRING]，Required	"encoder" 或 "decoder" 指定要切换到 encoder 或 decoder 模式。

Response

Example:

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

注意：

切换 Encoder/Decoder 模式需要等待较长的时间，因此，如果您需要特别处理 HTTP 请求/响应的超时，请注意该请求的执行时间最长可能需要 5 秒。所以您需要特别设置该 API 的请求超时，以免因为过短的超时而导致 API 请求意外失败。

如果您需要完全确认 Encoder/Decoder 模式是否正常工作，即使调用本 API 切换成功返回后，您还需要进一步尝试调用 /api/mode/status.json 来确认 NDI Device 的工作状态。这里之所以看上去这么奇怪，是因为 Encoder/Decoder 模式的切换需要设备内部执行一次快速的 Reboot 动作。

2.3 查询编码/解码模式是否就绪

当您执行了 Encoder/Decoder 模式的切换操作后，由于设备内部执行了一次快速的 Reboot 动作，如果您需要确认切换的模式是否正常进入工作状态，你

可能需要周期性地持续调用本 API，直到返回的结果告诉您它已经进入工作模式为止。通常，从切换到进入工作状态，它可能需要 10~15 秒。

您不必担心，您当前的 Time stamp 和 Authorization Token 在切换编码/解码模式、设备内部执行快速 Reboot 后，它们仍然有效。

API URL

/api/mode/status.json

Request

Method: **GET/POST**

Parameters: NONE

Response

如果编码/解码模式工作就绪，将返回：

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

如果模式尚未处于就绪状态，将返回错误：

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

3. NDI Encoder 状态和设置

注意：本模块的 APIs 仅当设备工作于 Encoder 模式下才有效。

3.1 获取 NDI Encoding 配置

API URL

/api/encoder/ndi/get_config.json

Request

Method: **GET/POST**

参数	Value	说明
types	[STRING]，Required	"ndihx" 或 "ndifull" 指定获取 NDI 或 NDI|HX 的信息。

Response

Example:

```
{
    "result": "ok",
    "data": {
        "device_group": "",
        "netmask": "255.255.0.0",
        "resolution": "none",
        "resolution_list": [
```

```
            {
                "name": "1920x1080P 60Hz",
                "value": "1080P60"
            },
            {
                "name": "1920x1080P 50Hz",
                "value": "1080P50"
            }
        ],
        "netprefix": "239.255.0.0",
        "channel_name": "Channel-HX",
        "ttl": 127,
        "quality": 2000,
        "ndi_connection": "tcp",
        "video_source":"HDMI",
        "codec":"H265",
        "hx_version":3
    }
}
```


Data 字段说明：

Field	Value	说明
device_group	[STRING]	NDI Group Name。如果该值为 ""（空字符串），表示启用 NDI 默认的组，即 public
channel_name	[STRING]	NDI 编码通道名称。

Field	Value	说明
ndi_connection	[STRING]	"tcp" 或 "multicast" 或 "no_multi_tcp"。 作为 NDI Sender，建议 NDI Receiver 采用的 Connection 方式。按 NewTek 的建议，目前允许 tcp（默认）或 UDP multicast 方式。
netprefix	[STRING]	*(仅当 ndi_connection 为 "multicast" 时有意义)* A multicast IP address (224.0.0.0 ~ 239.255.255.255) netprefix 和 netmask 是成对使用的关系。和我们通常的 IP 和 netmask 关系相似，由 netmask 决定 NDI multicast 地址的 Subnet。NDI SDK 将在这个 Subnet 中随机选择 Multicast 地址用于实际的 NDI 传输。
netmask	[STRING]	*(仅当 ndi_connection 为"multicast"时有意义)* The netmask for multicast IP address
quality	[INT]	值有效范围 2000 ~ 40000 目前只有 NDIHX 有此配置项，FULLNDI 此版本没有该配置项
ttl	[INT]	生存时间
resolution	[STRING]	当前选择的缩放,为"none"时不进行缩放
resolution_list	[OBJECT]	可支持的缩放列表，设置时用列表中的 value 值进行请求
video_source	[STRING]	当前视频来源 "HDMI"/"SDI" 或 "USB"
codec	[STRING]	视频编码格式，"H265" 或 "H264",仅当 types 为 "ndihx" 时有效
hx_version	[INT]	NDIHX 版本，2 或 3,仅当 types 为 "ndihx" 时有效

请注意：当前版本的 HTTP API 可能在返回结果中包含某些其它的未列在文档中的字段。这些字段的存在是因为出于软件兼容性的目的，请忽略它们。在后续的软件升级中，未列于文档中的字段很有可能会发生改变或丢弃。

3.2 设置 NDI Encoding 参数

API URL

/api/encoder/ndi/set_config.json

Request

Method: **GET/POST**

Parameter	Value	说明
types	[STRING]	有效值为"ndihx"或"ndifull"，分别表示修改 NDIHX 和 NDI 的信息
device_group	[STRING], Optional	当您指定了 device_group, device_name, channel_name 其中的任意一个或多个参数时，它将修改 NDI 的 Group, Device name 或 Channel Name。没有指定的参数，将保留之前设定的值。
channel_name	[STRING], Optional	通道名，注意:NDI 和 NDI|HX 的通道名不能一样
ndi_connection	[STRING], Optional	有效值为"tcp"或"multicast"或"no_multi_tcp"，指定建议 NDI Receiver 采用的 Connection 方式。当指定为"multicast"时，您还应该同时指定 netprefix 和 netmask 参数。
netprefix	[STRING], Optional	netprefix 和 netmask 是成对使用的关系。和我们通常的 IP 和 netmask 关系相似，由 netmask 决定 NDI multicast 地址的 Subnet。NDI SDK 将在这个 Subnet 中随机选择 Multicast 地址用于实际的 NDI 传输。
netmask	[STRING], Optional	如果您设置 net_connection="multicast" 但没有指

Parameter	Value	说明
		定 netprefix/netmask，如果之前有设置 netprefix/netmask 的值，则将沿用之前设置的值 ；否则，NDI Device 将随机生成一个 Multicast 地址，并选择 255.255.0.0 作为 netmask。
quality	[INT], Optional	编码质量，只在 types=ndihx 时生效，范围 2000-40000
ttl	[INT], Optional	生存时间
resolution	[STRING], Optional	缩放，当值为"none"时不进行缩放，否则传递合适的 value 值，value 值参考 3.1 的 resolution_list
codec	[STRING], Optional	参考值 ”H265“ 或 ”H264“,仅当 types 为"ndihx"时有效
hx_version	[INT], Optional	NDIHX 版本,参考值 2 或 3,仅当 types 为"ndihx"时有效

Response

Example:z

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

请注意：

- 如果 NDI Encoding 参数的**值发生了改变**，NDI Device 将断开当前的 NDI Connections、重新配置 NDI Sender 并再次启动。这个过程通常很快，大多数 NDI Receivers 也能在瞬间获得重新连接。但您必须注意，**它会重置连接到当前 NDI Device 的所有 NDI Connections**。

- 上面列出的所有参数都是可选的。如果您没有指定任何参数，意味着它什么也不做。

3.3 获取 NDI Encoding 状态和视频/音频格式信息

API URL

/api/encoder/ndi/status.json

Request

Method: **GET/POST**

参数	Value	说明
types	[STRING]，Required	"ndihx" 或 "ndifull" 指定获取 NDI 或 NDI|HX 的信息。

Response

Example:


```
{
    "result": "ok",
    "data": {
        "audio_sampling": 48000,
        "audio_format": "AAC",
        "audio_channels": 2,
        "audio_signal": "embedded",
        "samp_chan": "48KHz/2ch",
```

```
            "video_signal": false,
            "yRes": 1080,
            "xRes": 1920,
            "frame_rate": 30,
            "bitrate": 383222,
            "video_source": "HDMI",
            "interlaced": false,
            "serial_number": "1233211234567",
            "resolution": "1920x1080@30",
            "mode_version": "H.264 NDI|HX2",
            "src_resolution": "1920x1080@60",
            "src_frame_rate": 60,
            "src_xRes": 1920,
            "src_yRes": 1080
        }
}
```


Data 字段说明：

Field	Value	说明
video_signal	[BOOLEAN]	video_signal 用于指示当前 NDI encoding 视频输入的信号状态。当 video_signal=**false** 时，表示目前没有从视频输入接口获得视频输入（或者视频格式不被识别/支持），为 **true** 时，表示有信号
resolution	[STRING]	视频分辨率名称。 请注意：这是一个用户友好的分辨率名称，也许它不适合你的程序获取视频的详细格式信息。您可以通过 **xRes / yRes / frame_rate /**

Field	Value	说明
		interlaced 字段来获得程序友好的分辨率信息。
xRes	[INT]	xRes/yRes 表示当前视频的宽/高，以像素为单位。如果 xRes/yRes = 0，则表示当前分辨率无效。
yRes	[INT]	
frame_rate	[NUMBER]	Frame rate，单位：fps。如果 frame_rate = 0，则表示当前视频的刷新率无效。
interlaced	[BOOLEAN]	如果当前视频是 interlaced 格式，则值为 true；否则为 false
bitrate	[INT]	当前 NDI 编码（视频）的实时码率，Kbps
audio_format	[STRING]	**音频格式。**请注意：这是一个用户友好的音频格式名称（字符串），也许它不适合你的程序获取音频的详细格式信息。您可以通过 audio_sampling / audio_channels 来获得程序友好的音频信息。
audio_signal	[STRING]	none：无音频输入 embedded：来自于 HDMI/SDI 的内嵌数字音频 analog：来自于模拟音频输入 请注意：对于不同的设备，embedded 所代表的音频来源是有差异的。对于 HDMI Encoder，这表示音频来自于 HDMI 的内嵌数字音频；对于 SDI Encoder，这表示来自于 SDI 内嵌数字音频。 同样，analog 对于不同的设备也有不同的意义。有的设备模拟音频的输入为 Line In；而有的设备可能是 Microphone In，请注意区分。
audio_sampling	[INT]	当前音频的输入采样率
audio_channels	[INT]	当前音频的声道数
serial_number	[STRING]	序列号，每个设备唯一标识
video_source	[STRING]	视频输入的方式，当前值固定为 HDMI
mode_version	[STRING]	视频格式

Field	Value	说明
src_resolution	[STRING]	信号源视频分辨率
src_frame_rate	[INT]	信号源视频帧率
src_xRes	[INT]	信号源视频宽
src_yRes	[INT]	信号源视频高

3.4 获取视频编码视频源列表

API URL

/api/encoder/ndi/get_video_source.json

Request

Method: **GET/POST**

Response

Example:

```
{
    "result":"ok",
        "data":["HDMI","USB"]
}
```

Data 字段说明：

Field	Value	说明
DATA	[ARRAY]	可选视频源列表 "HDMI","SDI","USB",注意:USB 仅当插入 USB 视频才会在列表中

3.5 设置视频编码视频源

API URL

/api/encoder/ndi/select_video_source.json

Request

Method: **GET/POST**

参数	Value	说明
video_source	[STRING]	选择的音频源,参数参考 3.4 接口返回的 data

Response

Example:

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

3.6 获取音频信号源和音量

API URL

/api/audio/get_audio.json

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "signal": "HDMI",
        "volume": 100,
        "channels": 2

    }
}
```

Data 字段说明：

Field	Value	说明
signal	[STRING]	HDMI/SDI 来自于 HDMI/SDI 的内嵌数字音频 Line In 来自于模拟音频输入 USB 来自于 USB 视频输入的音频

Field	Value	说明
volume	[INT]	当前的音频音量，有效值范围 0 ~ 200。100 表示原始的音频增益；<100 表示音量减小；>100 表示音量增大。
channels	[INT]	音频声道数（目前产品只有两声道选项）

3.7 设置信号源和音量

API URL

/api/audio/set_audio.json

Request

Method: **GET/POST**

Parameter	Value	说明
signal	[STRING], Optional	HDMI/SDI 来自于 HDMI/SDI 的内嵌数字音频 Line In 来自于模拟音频输入 USB 来自于 USB 视频输入的音频 选择当前的音频输入。请参考 3.4 关于音频 signal 的说明。 如果不指定，则不修改当前的音频输入。
volume	[INT], Optional	调节音频输入的音量。有效值范围 0 ~ 200。100 表示原始的音频增益；<100 表示音量减小；>100 表示音量增大。 如果不指定，则不修改当前的音频音量。

Response

Example:

```
{
```

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

3.8 获取 NDI 开启关闭状态(只对 FULL NDI 有效)

API URL

/api/encoder/ndi/get_NDI_enable.json

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result": "ok",
    "data": {
        "enable": true
    }
}
```

Data 字段说明：

Field	Value	说明
enable	[BOOLEAN]	当值为 true 时，表示 FULL NDI 开启状态，为 false 时表示 FULL NDI 未开启

3.9 设置 NDI 开启关闭状态(只对 FULL NDI 有效)

API URL

/api/encoder/ndi/set_NDI_enable.json

Request

Method: **GET/POST**

Parameter	Value	说明
enable	[BOOLEAN], Optional	当值为 true 时，表示开启 FULL NDI，为 false 时表示关闭 FULL NDI

Response

Example:

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

4. NDI Sources 发现

Module name: decoder/discovery

Basic URL: /api/decoder/discovery/

注意：本模块的 APIs 仅当设备工作于 Decoder 模式下才有效。

4.1 获取网络中发现的 NDI Sources

API URL

/api/decoder/discovery/get.json

Request

Method: **GET/POST**

Parameter	Value	说明
force	[ANY], Optional	如果您指定了 Force 参数，无论它是什么值，代表要求 NDI Device 强制重新扫描网络。

Response

Example:

```
{
    "result": "ok",
    "data": [
        {
            "enable": 1,
            "id": "1",
```

```
            "device_name": "E1_NDI-1212-111",
            "url": "192.168.2.170:5963",
            "series": "NDI",
            "group_name": "public",
            "ip": "192.168.2.170",
            "group": "public",
            "channel_name": "Chan 1",
            "original_url": "192.168.2.170:5963",
            "name": "E1_NDI-1212-111 (Chan 1)",
            "port": 5963
        },
        /* ... Each discovered items in the array ... */
    ],
    "data_size": 10
}
```


Data 字段说明：

"data"是一个 JSON 数组，"data_size"是一个辅助描述"data"数组大小的字段（也许你用不上它）。

数组中的每一个 item 代表一个网络中发现的 NDI Source:

Field	Value	说明
group	[STRING]	NDI Source 所在的 NDI Group。如果您得到的 Group 是空字符串""，按照 NDI 的规则，它表示默认 public group。
name	[STRING]	NDI Source Name。这是 NDI Source 原始的名称。如果您想获得设备名称、通道名称，请参考 device_name 和 channel_name。

Field	Value	说明
device_name	[STRING]	从 name 中解析出来的 NDI Device Name，通常它也代表了 NDI Source 的 host name。
channel_name	[STRING]	从 name 中解析出来的 NDI Channel Name
url	[STRING]	NDI Source URL。**请注意：**目前 URL 的格式为 <IP>:<port>，但是建议您不要认为它就是事实，NDI SDK 并不担保这一点。在使用时，请保留 url 的原始值，不要尝试改变它。
enable	[INT]	1: 这个 Preset 有定义 0: 这个 Preset 没有定义 NDI Source；在这种情况下，所有后续的字段都没有意义
id	[INT]	序号
series	[STRING]	NDI 源
group_name	[STRING]	NDI Source 所在的 NDI Group。如果您得到的 Group 是空字符串""，按照 NDI 的规则，它表示默认 public group。
ip	[STRING]	ip 地址
original_url	[STRING]	NDI Source URL。**请注意：**目前 URL 的格式为 <IP>:<port>，但是建议您不要认为它就是事实，NDI SDK 并不担保这一点。在使用时，请保留 url 的原始值，不要尝试改变它。
port	[STRING]	端口号

TIPS:

A. NDI Discovery 基于 mDNS 机制，在默认情况下，它只能发现与 NDI Device 在同一子网之内的 NDI Sources。

B. 考虑到您的网络中可能存在大量的 NDI Sources，由于 NDI Discovery 的速度不如您想象的那么快，所以 NDI Device 会 cache 之前发现的 sources，并尽可能快地把结果返回给您。与此同时，NDI Device 会持续在后台扫描并更新

cache。所以，您得到的结果或许不完全是最新的（但是绝大多数情况下是）。您可以通过周期性地调用这个 API 来获得更新。

4.2 手动发现：指定 IP 或 NDI Groups

大多数时候，我们可以通过网络自动发现 NDI Sources。但有一些特殊情况：

- NDI 自动发现无法发现不在同一 Subnet 之内的 NDI Sources；
- NDI 有逻辑分组，默认情况下，NDI 自动发现只发现 NDI 分组为 public 的设备；
- 其它因为网络策略而限制了网络组播(Multicast)的情况。

在这些情况下，我们可能需要手动指定 NDI 发现的目标 IP 地址和/或 NDI Groups。

API URL

/api/decoder/discovery/set_manual_targets.json

Request

Method: **POST** （请注意本 API 只能使用 POST 方法，且提交的参数需要使用 JSON 格式）

Parameters (example):

```
{
    "ip": ["192.168.100.3", "172.16.10.5" /*, ...*/],
```

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

参数	Value	说明
ip	[ARRAY(STRING)]，Optional	通过数组的方式，指定一个或多个手动发现 NDI Sources 的目标 IP 地址。指定的 IP 地址可以与当前 NDI Device 不在同一 Subnet 之中。 该参数可选。如果没有指定，**将清除**之前指定的 Manual IPs。
group_name	[ARRAY(STRING)]，Optional	通过数组的方式，指定一个或多个手动发现 NDI Sources 的 NDI Group Names。 该参数可选。如果没有指定，**将清除**之前指定的 Group Names。

请注意：

A) 您手动指定的 IP 和 NDI Groups 将会被设备记录和保存。因此，即使是您重新启动了设备，这些手动指定的 IP 和 NDI Groups 都会起作用。

B) 手动指定 IP 和 NDI Groups，不会影响自动发现功能。因此，在发现结果列表中，您将看到所有自动发现的 NDI Sources 以及手动指定的 Sources（如果目标存在的话）。

C) 如参数说明中所提示的，如果您不指定 "ip" 或 "group_name"，意味着它会清除之前的设置。如果您需要保留之前的设置，请务必将原先的值作为参数传递。您可以通过 API **/api/decoder/discovery/get_manual_targets.json** 获取之前设置的值 。

Response

Example:

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

4.3 获得指定的手动发现 IP 和 NDI Groups

API URL

/api/decoder/discovery/get_manual_targets.json

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 字段说明：

Field	Value	说明
ip	[ARRAY(STRING)]	通过数组返回所有手动指定的 IP 地址列表。
group_name	[ARRAY(STRING)]	通过数组返回所有手动指定的 NDI Groups。

5. NDI 解码： Preset

我们的 NDI 解码设备有一个非常实用的功能：Preset。Preset 所表示的意义是：您可以将发现的 NDI Source 添加到由 1 ~ 9 所代表的 9 个 Presets 中，这是您的"收藏夹(Favorites)"。当您需要快速选择添加在 Preset 中的 NDI Source 进行解码时，您只需要指定 Preset 的 ID (1~9)即可。

Preset 可以为我们的 NDI 解码设备带来一些额外的功能优势，比如我们会在将来支持外接物理键盘，通过按键盘上的 1~9 数字键来完成快速切换。

NDI 解码设备上还有一个特殊的 Preset - 0。这个 Preset 代表的意义为 Blank，用于为屏幕填充一个可自定义的颜色。

Module name: decoder/preset

Basic URL: /api/decoder/preset/

5.1 获得当前的 Preset 列表和状态

API URL

/api/decoder/preset/status.json

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",
            "group_name":"public",
            "channel_name": "Channel 1",
            "url": "192.168.100.168:5961",
            "ip": "192.168.100.168",
```

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

    "data_size": 10
}
```

Data 字段说明：

"data"是一个 JSON 数组，"data_size"是一个辅助描述"data"数组大小的字段（也许你用不上它）。

数组中的每一个 item 代表一个 Preset:

Field	Value	说明
id	[STRING]	1 ~ 9 代表常规的 NDI Source Preset；0 代表 Blank
enable	[INT]	1：这个 Preset 有定义 0：这个 Preset 没有定义 NDI Source；在这种情况下，所有后续的字段都没有意义
group	[STRING]	Preset 所定义的 NDI Source 的 Group name
name	[STRING]	Preset 所定义的 NDI Source 的名称
device_name	[STRING]	从 `name` 字段中提取出来的 NDI Source 的 Device Name

Field	Value	说明
channel_name	[STRING]	从 name 字段中提取出来的 NDI Source 的 Channel Name
url	[STRING]	NDI Source 的原始 URL
ip	[STRING]	从 url 字段中提出来的 NDI Source 的 IP 地址
online	[STRING]	根据网络 NDI 发现的结果，检测出这个 Preset 所定义的 NDI Source 的在线状态: on: 当前 NDI Source 在线（可被发现） off: 当前 NDI Source 不在线（不能发现） **请注意**: online 状态是由 NDI Discovery 的结果而决定的，并不代表 NDI Source 是否可以连接和获取视频数据。所以，它只提供参考意义，不能作为设备确定在线的依据。
current	[BOOLEAN]	true: 这个 Preset 是当前正在解码的 NDI Source false: 这个 Preset 不是正在解码的 NDI Source
color	[STRING]	*仅对于 id = "0" 有效。* 对于 Preset 0 (Blank)，它不包括 enable, group, name, device_name, channel_name, url, ip, online 这些字段，而返回 color 代表当前的 Blank 填充颜色。具体格式请参见 6.4 /api/decoder/preset/set_blank 中的描述。

5.2 添加 NDI Source 到一个指定的 Preset

API URL

/api/decoder/preset/add.json

Request

Method: **GET/POST**

Parameter	Value	说明
position	[STRING], Required	指定 Preset ID，有效范围 1 ~ 9
name	[STRING], Required	NDI Source Name，您可以通过 API /api/decoder/discovery/get.json 获取发现的 NDI Sources 列表，并从中提取 name 字段
url	[STRING], Required	NDI Source 原始的 URL，您可以通过 API /api/decoder/discovery/get.json 获取发现的 NDI Sources 列表，并从中提取 url 字段
group	[STRING], Optional	NDI Source 所在的 NDI group name。 这个参数是可选的。但如果您能通过 API /api/decoder/discovery/get.json 获取并从中提取 group 字段，作为参数传递进来，将最好不过。

请注意：如果您指定的 Preset ID 原先已有 NDI Source 定义，那么新添加的 NDI Source 将会覆盖老的定义。

Response

Example:

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

5.3 移除 Preset 的 NDI Source 定义

API URL

/api/decoder/preset/remove.json

Request

Method: **GET/POST**

Parameter	Value	说明
id	[STRING], Required	指定要移除的 Preset ID，有效范围 1 ~ 9

Response

Example:

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

5.4 定义 Preset 0 (Blank)的颜色

API URL

/api/decoder/preset/set_blank.json

Request

Method: **GET/POST**

Parameter	Value	说明
color	[STRING], Required	指定 Preset 0 (Blank)的颜色。颜色的描述格式与 HTML 中对颜色的描述是保持一致的，即： 1. 使用 RGBA 色彩 2. 以#开头、十六进

Parameter	Value	说明
		制分别描述 R、G、B 和 A 我们常见的色彩描述形式： #RGB（例如 #f00 表示红色） #RRGGBB（例如 #ff00ff 表示紫色） #RRGGBBAA（例如 #00ff0080 表示绿色、50% 透明度）**请注意：**目前的版本不支持 Alpha 功能。

Response

Example:

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

6. 解码输出

Module name: decoder/current

Basic URL: /api/decoder/current/

6.1 获取当前解码状态

API URL

/api/decoder/current/status.json

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:


```
{
    "result": "ok",
    "data": {
        "isFull": false,
        "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": "NDI-FULL",
        "xRes": 1920,
        "yRes": 1080,
        "frame_rate": 59.94,
        "inst_frame_rate": 59.93,
        "bitrate": 125000,
        "audio_format": "48KHz / 2CH",
        "audio_sampling": 48000,
        "audio_channels": 2,
        "warning": "",
        "ptz":1,
        "video_mode":"hdmi"
```

```
    }
}
```

Data 字段说明：

Field	Value	说明
isFull	[BOOLEAN]	是否是 full ndi
name	[STRING]	当前正在解码的 NDI Source Name。如果当前正在输出 Preset 0 (Blank)，则 name = "(Blank)"
url	[STRING]	当前正在解码的 NDI Source 的原始 URL。如果没有解码 NDI Source，则 url = "" (空字符串)
ip	[STRING]	当前正在解码的 NDI Source 的 IP 地址。如果没有解码，则 ip = "0.0.0.0"
online	[BOOLEAN]	true: 当前 NDI Source 在线； false: 当前 NDI Source 不在线
resolution	[STRING]	当前 NDI Source 的视频分辨率（友好名称）。如果您需要获取视频的程序友好的信息，例如宽度/高度等，请参考 xRes, yRes, frame_rate, interlaced 等字段。
codec	[STRING]	当前 NDI Source 的视频 CODEC ID: **H264**: ndi hx **NDI-FULL**: full ndi 其它 CODEC ID 可能会存在。
xRes	[INT]	当前 NDI Source 的视频分辨率宽度
yRes	[INT]	当前 NDI Source 的视频分辨率高度
frame_rate	[NUMBER]	当前 NDI Source 的帧率。 **请注意：** frame_rate 指的是 NDI Source 的原始的 frame_rate。另一个字段 inst_frame_rate 表示(Instant Frame Rate，实时的帧率)，inst_frame_rate 表示的是由程序统计的、实际的帧率。
inst_frame_rate	[NUMBER]	Instant Frame Rate，由程序统计的、实际的帧率

Field	Value	说明
bitrate	[INT]	当前实时统计的 bitrate，Kbps
audio_format	[STRING]	当前音频格式（友好名称）。如果您需要获取音频的程序友好的信息，请参考 audio_sampling， audio_channels
audio_sampling	[INT]	当前音频输出的采样率
audio_channels	[INT]	当前音频输出的声道数
warning	[STRING]	目前已定义如下 string ID（**粗体字**）用于描述在解码输出时发生的警告/问题： **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
ptz	[STRING]	ptz 图标标识 其中 1 为显示，0 为隐藏
video_mode	[STRING]	音视频信号 目前多为 hdmi

6.2 将 Preset 或指定的 NDI Source 切换到当前解码输出

API URL

/api/decoder/current/set.json

Request

Method: **GET/POST**

Parameter	Value	说明
id	[STRING], Optional	指定 Preset ID (0~9)，将 Preset ID 所保存的 NDI Source 切换到当前解码输出。这个参数是可选的，如果指定了 id，那么下面的其它参数将没有意义；如果没有指定 id，则您必须指定下面的 name 和 url 参数。 如果您指定 id = "0"，表示当前不解码而直接输出 Blank。请参见 5 关于 Preset 0 的描述。
name	[STRING], Optional	如果您没有指定上面的 id 参数，那么 name 和 url 参数是必须的；否则无需指定。 name 是 NDI Source 的名称，通常您可以通过 4.1 /api/decoder/discovery/get.json 获得。根据 NDI SDK 的现有规则，url 是获得 NDI Source 连接的关键参数，name 仅提供发现 NDI source 的参考。所以，某种意义上说，您可以忽略 name 或给任意字符串。**但我们强烈建议**您应该提供正确和合法的 name。
url	[STRING], Optional	NDI source 的原始 URL。如果您没有指定参数 id，那么 url 参数是必须的。

Response

Example:

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

7. 解码输出高级选项

7.1 获取可配置的分辨率列表

此接口为 7.2 接口提供可配置的分辨率选项

API URL

/api/decoder/output/resolutions.json

Request

Response

Example:

```
{
    "result": "ok",
    "data": [
        {
            "name": "1920x1080P 60Hz",
            "value": "1080P60"
        },
        {
            "name": "1920x1080P 50Hz",
            "value": "1080P50"
        }
        /*...*/
    ]
}
```

7.2 设置视频/音频输出的高级选项

NDI Decoder 输出视频/音频的默认规则是跟随 NDI Source 的格式，即输出的视频分辨率/帧率、音频的采样率/声道数和 NDI Source 保持一致。

但是您可以指定视频输出分辨率、帧率、音频采样率的高级选项。

API URL

/api/decoder/output/set.json

Request

Method: **GET/POST**

Parameter	Value	说明
resolution	[STRING], Optional	用于配置解码输出的分辨率 可根据 7.1/api/decoder/output/resolutions.json 接口获取可配置的分辨率列表，然后将其中的 value 值作为此参数传入
channels	[INT], Optional	设置音频通道 **注意：** 由于当前版本硬件的能力限制，channels 只允许设置为 2。所以，目前请忽略该参数。
audio_format	[INT], Optional	指音频的输出采样率。 **注意：** 由于当前版本硬件的能力限制，sample_rate 只允许设置为 48000。所以，目前请忽略该参数。

Response

Example:

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

7.3 获取视频/音频输出的高级选项

API URL

/api/decoder/output/get.json

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:


```
{
    "result": "ok",
    "data": {
        "enable":       1,
        "frameRate":    0,
        "audio_format": 48000,
        "hdmi_mode":"hdmi",
        "resolution": "1080P50",
        "frame_rate": 59.94,
        "sample_rate": 48000,
        "colorspace":"auto",
```

```
        "channels":2
    }
}
```

Field	Value	说明
resolution	[STRING]	参见 7.2 /api/decoder/output/set.json 说明
channels	[INT]	参见 7.2 /api/decoder/output/set.json 说明
audio_format	[INT]	参见 7.2 /api/decoder/output/set.json 说明

8. Tally 状态和控制

Module name: tally

Basic URL: /api/tally/

Tally 状态和控制对于 Decoder 模式有作用。在 Decoder 模式下，您设置的 Tally 状态(Program On/Off，Preview On/Off)将会发送给它正在解码的 NDI Source。

8.1 获取当前 Tally 状态

API URL

/api/tally/status.json

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

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

Data 字段说明：

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

8.2 设置当前 Tally 状态

API URL

/api/tally/set.json

Request

Method: **GET/POST**

Parameter	Value	说明
pgm	[INT], Optional	1：设置 Program On 0：设置 Program Off 未指定：保持之前的状态
pvw	[INT], Optional	1：设置 Preview On 0：设置 Preview Off 未指定：保持之前的状态

Response

Example:

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

注意：

- 如果 NDI Device 工作在 Decoder 模式，您所设置的 Tally 状态将会发送到它正在解码的 NDI Source；当您切换 NDI Source 时，它会清除之前 NDI Source 的 Tally 状态，并且将状态发送到新的 NDI Source。

9. 系统时间

Module name: sys-time

Basic URL: /api/sys-time/

9.1 获取当前系统时间/配置

API URL

/api/sys-time/get.json

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

```
{
    "result":"ok",
    "data":{
        "ntp":"ntp1.aliyun.com ntp2.aliyun.com ntp3.aliyun.com",
        "Timezone":"Asia\/Shanghai",
        "time":"2023-02-28 16:50:50",
        "timetype":"pc"
        }
}
```

Data 字段说明：

Field	Value	说明
ntp	[STRING]	ntp 服务器地址，多个地址用空格间隔
Timezone	[STRING]	时区
time	[STRING]	当前系统时间
timetype	[STRING]	当前授时方式 pc/ntp

9.2 设置当前系统时间/配置

API URL

/api/sys-time/set.json

Request

Method: **GET/POST**

Parameter	Value	说明
ntp	[STRING]	ntp 服务器地址，多个地址用空格间隔
Timezone	[STRING]	时区
time	[STRING]	当前系统时间(适用于 pc 校时)
timetype	[STRING]	当前授时方式 pc 或 ntp

Response

Example:

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

```
}
```

10. 系统管理和控制

Module name: sys

Basic URL: /api/sys/

10.1 获取设备的工作状态

API URL

/api/sys/server_info.json

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	说明
addr	[STRING]	当前设备的 IP 地址
port	[INT]	当前您请求 HTTP API 所访问的端口（默认 HTTP: 80, HTTPS: 443）
name	[STRING]	**注意：** 这是一个将被放弃或改变的字段。目前代表的是您访问 HTTP API 的 URL。
persis	[STRING]	设备持续工作的时间长度，格式为：<小时>H<分>M <秒>S
start_time	[STRING]	设备开始启动的时间，格式为：Year-Month-Day Hour:Minute:Second
cpu_cores	[INT]	当前设备的 CPU Core 数量
cpu_payload	[INT]	当前 CPU 的负载%
mem_used	[NUM]	当前使用的内存数量，MB
mem_total	[NUM]	当前设备总共可用的内存，MB

10.2 设备重新启动

本操作将控制 NDI 设备重新启动。

调用本 API，HTTP 将会立即返回，但设备真正的重新启动动作将在 API 执行成功后 3 秒开始，整个重启的过程大约需要 20 秒。由于设备的网络通常采用 DHCP 获取地址，而 DHCP 获取的时间取决您的实际网络条件，所以，您能预期的可以重新访问设备的时间将 >= 30 秒。

设备重启的过程中，您将无法再访问任何 HTTP API，直到设备重启完成。

特别注意：设备重新启动后，您现有的安全凭据（Session ID 和 Authorization Token）将失效，您必须按照第 2 节所描述的方法，重新进行 HTTP API 授权！

API URL

/api/sys/reboot.json

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

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

10.3 恢复出厂设置

本操作将恢复 NDI Device 的默认出厂设置。将影响的设置包括：

- 网络地址的获取方式将恢复为 DHCP；
- 网络的 Failsafe 地址将恢复为出厂默认值（192.168.100.168，或 192.168.1.168，取决于不同的设备，请参考产品使用说明手册）；
- NDI Group 将恢复为默认值(public)；
- NDI Device Name 将恢复为 "<PRODCT_TYPE>-<SERIAL-NUMBER>" 的格式；
- NDI Channel Name 将恢复为出厂的默认名称（通常是 Channel-1，但以实际产品为准）；
- NDI Connection 的方式将恢复为默认（TCP）连接方式；
- NDI Decoder 的当前解码 Source 和 Preset 将被清空，Preset 0 (Blank)的颜色将恢复为默认（黑色）；
- 所有的用户将被清除，admin 用户将恢复默认的密码。

调用本 API，HTTP 将会立即返回，但设备会在 API 执行成功后 3 秒开始重新启动，重新启动的行为和造成的影响同 `10.3` 所描述的一致。

API URL

/api/sys/restore.json

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

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

10.4 获取主机名

API URL

/api/device/get_hostname.json

Request

Method: **GET/POST**

Response

Example:

```
{
    "result": "ok",
    "data": {
        "hostname": "N6-123456"
```

```
    }
}
```

10.5 设置主机名

API URL

/api/device/set_hostname.json

Request

Method: **GET/POST**

Parameter	Value	说明
hostname	[STRING]	主机名

Response

Example:

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

注意：

修改主机名后，NDI 会重启以适应新的主机名

10.6 设置发现服务器

API URL

api/device/set_discovery_server.json

Request

Method: **GET/POST**

Parameter	Value	说明
address	[STRING]	发现服务器地址，多个发现服务器可以用",”间隔
enalbe	[BOOLEAN]	是否开启 true or false

Response

Example:

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

10.7 获取当前发现服务器状态

API URL

/api/device/get_discovery_server.json

Request

Method: **GET/POST**

Response

Example:

```
{
    "result": "ok",
    "data": {
        "address": "192.168.1.168",
        "enable": true
    }
}
```

10.8 获取当前系统状态

获取当前系统的 cpu 状态，内存状态，系统运行时间等状态

API URL

/api/sys/server_info.json

Request

Method: **GET/POST**

Response

Example:


```
{
    "result": "ok",
    "data": {
        "cpu_cores": 8,
        "cpu_payload": 22,
        "mem_used": 300.15,
        "addr": "192.168.40.161",
        "persis": "1H 20M 23S",
        "port": "443",
        "name": "https://192.168.40.161",
        "start_time": "2023-02-28 15:47:21",
        "mem_total": 1889.72
    }
}
```


Data 字段说明：

Field	Value	说明
cpu_cores	[INT]	cpu 核数
cpu_payload	[NUMBER]	CPU 使用率(百分比)
mem_total	[STRING]	内存总量(M)
mem_used	[STRING]	内存使用量(M)
addr	[STRING]	设备 ip
name	[STRING]	设备 web 地址
port	[STRING]	web https 端口

Field	Value	说明
persis	[STRING]	设备运行时间
start_time	[STRING]	设备开机时间

10.9 获取系统版本

API URL

/api/firmware/get.json

Request

Method: **GET/POST**

Response

Example:

```
{
    "result": "ok",
    "data": {
        "softwareVersion": "1.01.0001",
        "hardwareVersion": "1.0",
        "ndiVersion": "5.1.3"
    }
}
```

Field	Value	说明
softwareVersion	[STRING]	软件版本
hardwareVersion	[STRING]	硬件版本
ndiVersion	[STRING]	ndi 版本

11. 用户管理

Module name: users

Basic URL: /api/users/

11.1 获取当前用户列表

API URL

/api/users/list.json

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

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

```
    "data": [
        {
            "alias": "Admin",
            "api": true,
            "create_time": "1970-01-01 00:00:00",
            "id": "admin",
            "web": true
        }
    ]
}
```


Data 字段说明：

Field	Value	说明
alias	[STRING]	昵称
id	[STRING]	用户名
api	[BOOLEAN]	是否开放 api 权限
web	[BOOLEAN]	是否开放 web 登录权限
create_time	[STRING]	用户创建时间

11.2 添加用户

API URL

/api/users/add.json

Request

Method: **GET/POST**

Parameters:

Parameter	Value	说明
alias	[STRING]	昵称
id	[STRING]	用户名
api	[BOOLEAN]	是否开放 api 权限
web	[BOOLEAN]	是否开放 web 登录权限
password	[STRING]	密码

Response

Example:

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

11.3 修改用户信息

API URL

/api/users/modify.json

Request

Method: **GET/POST**

Parameters:

Parameter	Value	说明
id	[STRING]	用户名（必填，不能修改这个字段）
alias	[STRING]	昵称
api	[BOOLEAN]	是否开放 api 权限
web	[BOOLEAN]	是否开放 web 登录权限
password	[STRING]	密码

Response

Example:

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

11.4 删除用户

API URL

/api/users/remove.json

Request

Method: **GET/POST**

Parameters:

Parameter	Value	说明
id	[STRING] /[ARRAY(STRING)]	用户名(字符串或数组)

Response

Example:

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

12. 网络配置

Module name: network

Basic URL: /api/network/

12.1 获取当前网络配置

API URL

/api/network/get.json

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",
        }
    ],
    "data_size": 1
}
```


Data 字段说明：

Field	Value	说明
device	[STRING]	当前网络接口的设备名称。**请注意**这个设备名称，它是您修改网络配置的依据。也就是说，当您修改网络配置时，您必须指定这个 device 名称。

Field	Value	说明
state	[STRING]	当前网络的工作状态： up： 网络工作正常 down：由于网线没有接入而断开 disabled: 由于其它原因而禁止使用该网络 error: 网络工作故障
ip	[STRING]	当前网络配置的 IP 地址。**请注意：**如果您的网络配置的是 DHCP 地址获取，那么 ip 将显示当前 DHCP 实际获取的 IP 地址；而如果网络配置的是静态 IP 地址，则 ip 表示您所设置的静态 IP 地址
mask	[STRING]	当前网络的子网掩码 (netmask)
mac	[STRING]	当前网卡的 MAC 地址
gw	[STRING]	当前网络的默认网关(gateway)。如果没有设置，它将可能是""（空字符串）或者"0.0.0.0"
dynamic	[STRING]	y: 启动 DHCP 配置 n: 使用静态 IP 地址配置
dns	[STRING]	当前配置的 DNS 服务器地址。多个地址之间用 ';' 隔开。

12.2 修改网络配置

重要提示： 网络配置的修改是一个敏感操作，错误的配置可能导致设备无法正常访问。因此，我们提醒您注意：

- 常规的网络地址修改，请您尽量通过 NDI Device 的 Web UI 来完成；
- 如果因为修改网络参数而导致不可访问，您可以通过按设备的 Reset 按钮来恢复出厂设置。

API URL

/api/network/set.json

Request

Method: **GET/POST**

Parameter	Value	说明
device	[STRING], Required	指定要修改网络接口的名称。请参见 12.1 /api/network/get.json 中关于 device 参数的说明。这个参数是必须的。
dynamic	[STRING], Optional	y: IP 地址获取使用 DHCP。如果 dynamic="y"，则 ip, mask, gw, dns 参数将没有意义 n: 使用静态 IP 地址配置
ip	[STRING], Optional	配置 IP 地址
mask	[STRING], Optional	子网掩码 (netmask)
mac	[STRING], Optional	如果指定 mac，则可修改当前网卡的 MAC 地址。请注意： 1. 您必须提供有效、合法的 MAC 地址； 2. 除非您确定您要这样做的意图，否则请尽量不要修改设备的 MAC 地址
gw	[STRING], Optional	默认网关(gateway)。指定""（空字符串）表示无 gateway 配置。
dns	[STRING], Optional	指定 0 个或多个 DNS 服务器地址。多个地址之间用 ';' 隔开。""（空字符串）表示无 DNS 配置。

Response

Example:

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

注意：网络地址修改后，如果 IP 地址发生变化，您务须使用新的 IP 地址进行 HTTP API 访问。网络地址修改的生效通常 < 1s。但如果您指定 ip 地址为

DHCP 获取，取决于您的实际网络环境，它获得有效 IP 地址所需要等待的时间是不确定的。