NDI 设备HTTP API (Version 1.0)

NDI设备HTTP API允许您通过发送HTTP请求到特定的URL来管理和控制千视NDI嵌入式编码器和解码器设备，这包括N3、N4、N30、N40等。当前HTTP API的版本为1.0。

需要说明：目前的HTTP API不是严格意义上的RESTful API，它没有遵循RESTful API的形式化定义规则。因为我们认为对于设备管理和控制来说，RESTful的形式化定义并不能为我们带来特别的好处。无论如何，当前版本的HTTP API足够的简单、易于理解和使用。此外，如果您认为RESTful API对您来说是必要的，请随时告诉我们。

> **与RESTful API相比，NDI设备HTTP API (Version 1.0)有以下区别：**
>
> 1. 我们仅仅使用HTTP **GET/POST**请求，而不使用PUT/DELETE或其它请求；
> 2. 我们只使用JSON格式作为每一个API请求的返回结果（RESTful API大多数情况下也是如此）；
> 3. 每一个API请求的结果（成功或者失败的标识），我们通过JSON返回结果中的字段来标识，而非像RESTful API那样通过HTTP响应来标识。

要完整地支持本文档所描述的HTTP API，您需要将NDI设备的Firmware升级到主版本号为1.50或更高的版本。之前的版本可能无法完整支持这些API，或存在兼容性问题。

Index

1. 基本规则

1.1 URL规则

对于HTTP访问:

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

对于HTTPS访问:

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

Embedded NDI Devices均支持HTTP和HTTPS访问，这可以通过URL的http://或https://进行区分。HTTP的服务端口默认为80，HTTPS的服务端口默认为443。<host-ip>是对应您要请求的NDI设备的主机IP地址，这是必须的。

URL路径中，**/api/v1** 用于区分HTTP API的版本，随着后续的版本更新，这个地址可能会发生改变。

<module-name>是HTTP API对应的功能module名称，本文档将详细描述每一个module的细节；<module-name>可能是单一word，例如 user, tally, ...等等；也可能是以 / 分隔的路径形式，例如 decoder/preset, decoder/discovery, ...等等。详见文档中的具体描述。

<method-name>是对应module中的一个方法，例如：/api/v1/user/**add** ，add是user模块中的添加用户的方法。

1.2 HTTP请求

您可以通过HTTP GET或HTTP POST向对应的HTTP API methods提交请求。对于需要参数的methods，您可以有三种方法提交参数：

A) 通过HTTP GET方法，提交URL encoded参数

例如：

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

另外一个对特殊字符进行url encode的例子（将字符串"My device"中的空格进行url encode）：

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

B) 通过HTTP POST方法，提交application/x-www-form-urlencoded参数

这是常规的HTTP POST提交参数的方法。这意味着HTTP Request中的Content-Type指定为**application/x-www-form-urlencoded**（HTTP默认）。

例如：

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

pgm=1&pvw=0
```

另一个对特殊字符进行url encode的例子（将字符串"My device"中的空格进行url encode）：

```
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) 通过HTTP POST方法，提交JSON格式的object；Object中的每一个field对应一个参数

这要求您提交的HTTP请求中Content-Type为 **application/json**，而HTTP请求的Body为一个有效的JSON object。

例如：

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

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

另一个例子：

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

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

如果在本文档中没有特别声明，那么以上三种方式提交HTTP请求参数都是有效的，作用是等同的。

> 您可以使用**curl**、**Postman**等工具或者自己编写简单的javascript程序来完成HTTP API测试。

1.3 HTTP响应和错误处理

除非设备的Web server在工作时出现了异常，否则设备总会给予您HTTP **200 OK**响应，同时在HTTP Content中包含一个JSON object描述对应API执行的结果。

如果设备给予您非200 OK的响应，这表示Web server出现了工作异常，例如404错误表示无法找到对应的请求地址（URL错误）。请参考HTTP的错误代码、并进一步检测HTTP响应的错误消息获知详细的出错原因。

请注意：如果您的HTTP API请求地址正确，即使这个API执行失败，设备的Web server仍然会返回**200 OK**。您应该通过检查返回的JSON object中的字段来确定执行成功与否。**这是当前版本HTTP API与RESTful API在执行规则上的最大差异。**

如果HTTP API执行成功，返回的JSON object格式如下：

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

其中，**"result"** 是必定存在的字段，它的值**"ok"**表示请求的HTTP API执行成功。对于执行成功而言，**"msg"** 并非必定存在，您也可以完全忽略它的存在，它仅仅用于额外描述一些信息。对于有返回结果的HTTP API来说，**"data"** 字段将会存在，而且它是一个JSON **object** 或者 **array**，表示API的返回结果。这将视具体的API而定，请参考具体API的说明。

而如果HTTP API执行失败，返回的JSON object格式如下：

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

"result"是必定存在的字段，当前版本的HTTP API中，仅定义了**"error"**表示执行失败，以及**"auth-failed"**表示安全验证失败；但在未来的版本中，它可能会有其它的**非"ok"**的标识符号来代表更多的出错含义。如果"result"不为"ok"，您可以进一步检查**"msg"**来确定出错的原因。详见每一个API的说明。

HTTP响应的Content-Type必定为 application/json。

> **请注意：**在本文档的后续API描述内容中，如非特殊情况的必要说明，将忽略关于API请求错误的说明。

1.4 CHARSET

目前版本的HTTP API仅仅支持utf-8字符集。

1.5 超时

在您处理HTTP请求/响应的超时设置时，请注意：

- 绝大多数的API，从接受请求到完成执行，设备会在 <= 0.1s 时间内完成响应；但您需要根据实际的网络环境考虑传输延时等因素，合理设置超时值（一般来说，建议设置5秒超时或更高一些）。
- 但以下几个API您需要特别注意，它们的执行/响应时间应该特别设置，甚至需要进行特别的处理（详细情况请参考对应的API说明）：
    - /api/v1/sys/reconnect (重置NDI连接)
    - /api/v1/sys/reboot (重新启动设备)
    - /api/v1/sys/restore (恢复出厂设置)
    - /api/v1/mode/switch (在encoding和decoding模式之间切换，对于Connect Spark IO设备)
    - /api/v1/network/set (修改网络设置)

1.6 跨域请求(Cross-domain Request)

NDI Devices支持HTTP Cross-domain Request，但您需要遵循文档第2节所描述的安全性规则。

2. 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。

2.1 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**Session ID**和**Authorization Token**，并在响应中返回给您。

3) 您必须记录这对Session ID和Authorization Token。在您接下来的每一个HTTP API请求中，您都必须传递Session ID和Authorization Token在HTTP的请求Headers或参数中。这有三种可选择的传递方式：

- *通过HTTP GET/POST参数；*
- *通过HTTP Headers字段；*
- *通过HTTP Cookie。*

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 Session ID 和 Authorization Token，而是通过检查您提交的用户名和密码来为您生成Session ID和Authorization Token。

API URL

/api/v1/user/authorize

Request

Method：**GET/POST**

参数	Value	说明
username	**[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": {
        "session": "559dee0bd779a894618d6e044c35a3fc",
        "token": "4345cd7b092d762bd4a646a98aa9f8ff"
    }
}
```

Data字段说明：

Field	Value	说明
session	**[STRING]**	随机的Session ID。 Session ID和Authorization Token将在**您没有任何HTTP API操作之后的10分钟内失效。**
token	**[STRING]**	随机的Authorization Token。 您应该记录下session & token 的值，并在后续其它的HTTP API请求中传递这两个值。

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！

2.2 在HTTP API请求中传递Session ID和Authorization Token

如果您按照 2.1 所描述的方法请求授权成功，您将获得NDI Device为您生成的Session ID和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Session ID和Authorization Token是否合法。如果您没有提供Session ID和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": "...error message..."
}
```

有三种方法传递Session ID和Authorization Token：

A. [建议方法] 通过HTTP Request Headers传递

在HTTP Request Headers中，添加 **"API-Session"** Header field表示Session ID，以及添加 **"API-Token"** Header field表示Authorization Token。例如：

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

<BODY>
```

如果使用curl进行测试，命令如下：

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

B. 通过Cookie传递

通过HTTP Cookie字段 **"session"** 传递Session ID，以及字段 **"token"** 传递Authorization Token。例如：

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

<BODY>
```

如果使用curl进行测试，命令如下：

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

C. 通过GET/POST参数传递 [不建议]

如果方法 A 和方法 B 都不适用于您，那么您也可以通过 GET/POST 的参数来传递Session ID以及Authorization Token。但我们非常**不建议**您采用这种方式，因为这会破坏HTTP API参数的规范性，使得HTTP API的参数看起来混乱不堪。

在确有必要的情况下，您可以在HTTP GET/POST参数中传递 **"api-session"**作为Session ID，以及**"api-token"**作为Authorization Token。例如：

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

3. 编码和解码模式的状态及切换

Module name: mode

Basic URL: /api/v1/mode/

某些NDI设备同时具备Encoder和Decoder的能力（例如Connect Spark IO），您可以在这两者之间进行切换。但是Connect Spark Plus / TCMI4KUHD 只有Encoder的能力，在这种情况下，模式切换将不起作用。

3.1 获取当前编码/解码工作模式

API URL

/api/v1/mode/get

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

3.2 切换编码/解码工作模式

API URL

/api/v1/mode/switch

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"
}
```

如果切换失败，参见1.3中关于错误消息的描述。

注意：如果某些设备型号不支持encoder/decoder模式切换，本操作将不起作用，并返回相应的错误消息。

> 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/v1/mode/status 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。

3.3 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Session ID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/v1/mode/status

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

请注意：**"reason": "not-ready"** 不是任何时候都可能存在。简单地说，如果设备正在执行内部的Reboot过程中、出现程序临时性地无法访问时，**"reason"**字段将不会存在。但是，即使是Reboot的过程中，您的HTTP请求仍然可以被设备响应。

4. NDI Encoder状态和设置

Module name: encoder/ndi

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

注意：本模块的APIs仅当设备工作于Encoder模式下才有效。

4.1 获取NDI Encoding配置

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
device_name	**[STRING]**	NDI设备名称，它同时也代表着NDI Device的host name。
channel_name	**[STRING]**	NDI编码通道名称。
ndi_connection	**[STRING]**	**"tcp"** 或 **"multicast"**。 作为NDI Sender，建议NDI Receiver采用的Connection方式。按NDI的建议，目前允许tcp（默认）或 UDP multicast方式。UDP unicast方式将在未来提供支持。
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：75 ~ 150 由于历史原因，也许我们选择了一个不太恰当的参数名称，它所代表的意义是参照NDI标准推荐的bitrate，我们可以控制的bitrate in percent，范围在 75% ~ 150%之间。

> 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。

4.2 设置NDI Encoding参数

API URL

/api/v1/encoder/ndi/set_config

Request

Method: **GET/POST**

Parameter	Value	说明
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。
device_name	**[STRING], Optional**	
channel_name	**[STRING], Optional**	
ndi_connection	**[STRING], Optional**	有效值为**"tcp"**或**"multicast"**，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**	由于历史原因，也许我们选择了一个不太恰当的参数名称，它所代表的意义是参照NDI标准推荐的bitrate，我们可以控制的bitrate in percent，范围在 75% ~ 150%之间。

Response

Example:

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

如果设置成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

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。

4.3 获取NDI Encoding状态和视频/音频格式信息

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字段说明：

Field	Value	说明
video_signal	**[STRING]**	**none:** 没有视频信号输入 **hdmi:** 视频信号来自于HDMI，已识别输入格式 **sdi:** 视频信号来自于SDI，已识别输入格式 video_signal用于指示当前NDI encoding视频输入的信号状态。当video_signal=**"none"**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）。 "hdmi" / "sdi" 视不同的设备而定。
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 / 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

4.4 获取音频信号源和音量

API URL

/api/v1/encoder/ndi/get_audio

Request

Method: **GET/POST**

Parameters: NONE

Response

Example:

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

Data字段说明：

Field	Value	说明
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。
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（百分比）；>100表示音量增大（百分比）。

4.5 设置信号源和/或音量

API URL

/api/v1/encoder/ndi/set_audio

Request

Method: **GET/POST**

Parameter	Value	说明
signal	**[STRING], Optional**	**embedded**: 来自于HDMI/SDI的内嵌数字音频 **analog:** 来自于模拟音频输入 选择当前的音频输入。请参考 `4.4` 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（百分比）；>100表示音量增大（百分比）。 如果不指定，则不修改当前的音频音量。

Response

Example:

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

如果设置成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

5. NDI Sources发现

Module name: decoder/discovery

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

注意：本模块的APIs仅当设备工作于Decoder模式下才有效。

5.1 获取网络中发现的NDI Sources

API URL

/api/v1/decoder/discovery/get

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": [
        {
            "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。
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。

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。

5.2 手动发现：指定IP和/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/v1/decoder/discovery/set_manual_targets

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/v1/decoder/discovery/get_manual_targets** 获取之前设置的值 。

Response

Example:

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

如果设置成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

5.3 获得指定的手动发现IP和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。

6. 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/v1/decoder/preset/

6.1 获得当前的Preset列表和状态

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
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/v1/decoder/preset/set_blank` 中的描述。

6.2 添加NDI Source到一个指定的Preset

API URL

/api/v1/decoder/preset/add

Request

Method: **GET/POST**

Parameter	Value	说明
id	**[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/v1/decoder/discovery/get 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/v1/decoder/discovery/get 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/v1/decoder/discovery/get 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"
}
```

如果添加成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

6.3 移除Preset的NDI Source定义

API URL

/api/v1/decoder/preset/remove

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"
}
```

如果移除成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

6.4 定义Preset 0 (Blank)的颜色

API URL

/api/v1/decoder/preset/set_blank

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"
}
```

如果设置成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

7. 解码输出

Module name: decoder/current

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

7.1 获取当前解码状态

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字段说明：

Field	Value	说明
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： **SHQ0 / shq0**: NDI YUV4:2:0格式编码 **SHQ2 / shq2**: NDI YUV4:2:2格式编码 **SHQ7 / shq7**: NDI YUVA4:2:2:4格式编码 其它CODEC ID可能会存在，但止前暂不被支持。
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
interlaced	**[BOOLEAN]**	true: NDI Source是Interlaced视频 false: NDI Source是Progressive视频
de_interlace	**[BOOLEAN]**	NDI解码设备具有将interlaced视频执行Deinterlace并输出为Progressive格式的能力。这需要您通过 API `8.3 /api/v1/decoder/output/set` 进行设置（默认不执行Deinterlace操作）。 如果您配置了Deinterlace，而且当前NDI Source原始的视频是interlaced，那么 `de_interlace` 将会为true，它用于指示您，目前Decoder进行了Deinterlace转换。
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

Field	Value	说明
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

7.2 将Preset或指定的NDI Source切换到当前解码输出

API URL

/api/v1/decoder/current/set

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 `6` 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 5.1 `/api/v1/decoder/discovery/get` 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"
}
```

如果切换成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

8. 解码输出和切换的高级选项

在切换不同的NDI Source用于解码输出的时候，我们相信您一定希望这个切换是十分平滑和顺畅的，不能有任何的视频停滞、屏幕闪烁或其它不好的现象发生。

然而，我们要从技术的角度上告诉您一些事实：

- 从发起一个NDI Source的连接，到我们真正得到视频/音频数据，这个过程需要一些时间，也许需要0.1秒、0.2秒或者更长，这取决于您的网络环境。
- 如果切换到新的NDI Source，它的视频分辨率/帧率发生了变化，我们需要重新配置HDMI/SDI的输出格式。非常糟糕，这时候屏幕可能会短暂地没有输出或闪烁。

上面的情况，都会带给我们不好的体验。

我们尽最大的努力帮助您解决这些问题，在可能的情况下，我们会提供一些可设置的选项帮助您解决画面停滞、闪烁等问题。当然，NDI Decoder由于硬件能力的限制，目前的版本可能会有一些仍然无法解决的问题。

在以下情况下，NDI Decoder可以很好地解决画面停滞、闪烁的问题：

- 因为NDI连接需要等待时间而造成的停滞：通过 `8.1 /api/v1/decoder/output/set_smoothness` 设置平滑切换的等待超时可以解决；
- **在分辨率没有变化**而只是帧率变化的情况下：通过 `8.3 /api/v1/decoder/output/set` 强制指定输出的帧率可以解决；
- 通过 `8.3 /api/v1/decoder/output/set` 指定输出格式为deinterlace，可以将interlaced视频转换为progressive输出，以防止interlaced/progressive视频的变化而造成的显示问题。

在以下情况下，由于硬件能力的限制，目前还可能出现输出闪烁的情况：

- 分辨率发生改变

Module name: decoder/output

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

8.1 设置平滑切换的等待超时

为解决NDI连接需要等待时间而造成的画面停滞问题，我们采取了这样的策略：

当您发起新的NDI Source切换请求时，NDI Decoder会保持当前的解码不停止，同时尝试连接您指定的新的NDI Source。如果新的Source连接成功并得到了视频/音频数据，NDI Decoder会立即停止老的NDI Source解码、切换到新的NDI Source上。这样，可以让您的切换“如丝滑般顺畅”。

当然，真实的世界总是不尽如人意。有这么几种情况：

- 您的网络确实太糟糕了。新的NDI Source连接需要待很久很久的时间才能成功。如果NDI Decoder按照上面的方法一直输出老的NDI Source的内容，也许您会产生错觉，觉得您发起的NDI Source切换请求根本没有成功。
- 新的NDI Source完全无法连接。

为了应对这些情况，我们的策略是：您可以指定一个等待新的NDI Source连接成功的超时时间，如果Source在这个超时时间之内完成了连接，那么很好、这就是您所期望的；而如果在超时时间之内新的Source仍然无法连接，那么很不幸，我们需要强制停止当前的NDI Source解码输出，然后一直等待新的Source就绪。

如果您将这个超时设置为0，聪明的你一定想到了会发生什么——它会立即停止老的NDI Source，然后再重新连接新的Source。这将会导致出现画面停滞的现象。

API URL

/api/v1/decoder/output/set_smoothness

Request

Method: **GET/POST**

Parameter	Value	说明
timeout	**[INT], Required**	指定平滑切换的等待超时。单位：ms。 如果指定为0，表示不处理平滑切换。

Response

Example:

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

如果设置成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

8.2 获取平滑切换的等待超时

API URL

/api/v1/decoder/output/get_smoothness

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

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

Data字段说明：

Field	Value	说明
timeout	**[INT]**	当前的平滑切换等待超时，单位：ms

8.3 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/v1/decoder/output/set

Request

Method: **GET/POST**

Parameter	Value	说明
resolution	**[STRING], Optional**	目前您可以设置以下选项值： **auto**: 默认，跟随NDI Source的分辨率 **deint**: 如果NDI Source的分辨率是interlaced格式，将自动deinterlace到progressive格式（1080i -> 1080p转换）。如果Source的视频格式本身就是progressive，则不受任何影响。
frame_rate	**[INT], Optional**	指定视频输出的帧率。可选的值包括： **23.98, 24, 25, 29.97, 30, 50, 59.94, 60** 以及一个特殊的值: **0** 0表示使用NDI Source相同的帧率。 如果指定一个有效的帧率值，NDI Decoder会自动执行帧率转换。例如，NDI Source的原始视频格式是1920x1080p 60Hz，但如果指定frame_rate为29.97，NDI Decoder会自动转换为1920x1080p 29.97Hz输出。
sample_rate	**[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"
}
```

如果设置成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

8.4 获取视频/音频输出的高级选项

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	说明
resolution	**[STRING]**	参见 8.3 `/api/v1/decoder/output/set` 说明
frame_rate	**[INT]**	参见 8.3 `/api/v1/decoder/output/set` 说明
sample_rate	**[INT]**	参见 8.3 `/api/v1/decoder/output/set` 说明

9. Tally状态和控制

Module name: tally

Basic URL: /api/v1/tally/

Tally状态和控制对于 Encoder 和 Decoder 模式都有作用。在Encoder模式下，您可以通过API来获取当前的Tally状态，甚至可以由您来控制Tally灯的亮/灭（虽然看起来不是特别必要）；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。

9.1 获取当前Tally状态

API URL

/api/v1/tally/status

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

9.2 设置当前Tally状态

API URL

/api/v1/tally/set

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"
}
```

如果设置成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

注意：

- 如果当前NDI Device工作在Encoder模式，并且它正在被某个NDI Receiver连接，由于NDI Receiver可能更新Tally状态，所以Tally的设置也许会被更新的Tally状态覆盖。请注意这个细节。
- 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。

10. 系统管理和控制

Module name: sys

Basic URL: /api/v1/sys/

10.1 获取设备的工作状态

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	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 重置所有NDI连接

本操作将断开所有的NDI连接并重新开始：对于NDI Encoder而言，意味着它将重新初始化NDI Sender；对于NDI Decoder而言，它将断开现有的NDI Source并且重新连接。

完成重置的时间大约需要4-6秒，请注意。

API URL

/api/v1/sys/reconnect

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

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

如果执行成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

10.3 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/v1/sys/reboot

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

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

如果执行成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

10.4 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/v1/sys/restore

Request

Method: **GET/POST**

Parameters: **NONE**

Response

Example:

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

如果执行成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

11. 网络配置

Module name: network

Basic URL: /api/v1/network/

11.1 获取当前网络配置

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代表一个网络接口的配置:

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。
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。
ip2	**[STRING]**	设备的第二IP地址（也称之为Failsafe地址）。由于设备的主IP地址修改可能发生错误或因忘记而导致您无法访问设备，Failsafe地址允许您配置一个默认的、易于记忆的IP，此时您可以通过Failsafe地址来访问和配置设备。 所以，我们建议您尽可能不要修改ip2（Failsafe地址）。
mask2	**[STRING]**	ip2对应的子网掩码 (netmask)
gw2	**[STRING]**	ip2对应的默认网关 (gateway)。如果没有设置，它将可能是""（空字符串）或者"0.0.0.0"
ip3/mask3/gw3	**[STRING]**	ip3/mask3/gw3和ip2/mask2/gw2有相似的意义和用途。不过大多数情况下，您不必配置ip3。**我们不对此参数配置的有效性提供担保**

11.2 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/v1/network/set

Request

Method: **GET/POST**

Field	Value	说明
device	**[STRING], Required**	指定要修改网络接口的名称。请参见 11.1 `/api/v1/network/get` 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。
ip2	**[STRING], Optional**	配置Failsafe IP地址。如果您需要清除Failsafe地址，请设置 ip2 = ""（空字符串）
mask2	**[STRING], Optional**	Failsafe地址的子网掩码 (netmask)
gw2	**[STRING], Optional**	Failsafe地址对应的默认网关 (gateway)。指定""（空字符串）表示无gateway配置。
ip3/mask3/gw3	**[STRING], Optional**	ip3/mask3/gw3和ip2/mask2/gw2有相似的意义和用途。不过大多数情况下，您不必配置ip3。**我们不对此参数配置的有效性提供担保**

Response

Example:

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

如果执行成功，将返回 result = "ok" 的消息；否则请参见 1.3 的标准错误消息描述。

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。