Pro Convert Audio DX

前言

通用接口

设备状态

全局设置

直播

矩阵

dante

系统设置

网络设置

证书

用户

固件更新

系统日志

文档简介

针对**，我们开放了丰富的 API，方便开发人员与设备交互，如获取设备的基本信息（设备名称、固件版本等），修改设备配置，更新固件等。这些 API 基于 HTTP 协议，是一种轻量级、无连接状态的接口，响应数据为 JSON 格式。通过本文档，您可以更详细地了解每个 API 的功能和请求方式 。

本文档中的 API 可适用于以下产品：

- Pro Convert Audio DX

接口约定

一、概要

- 请求协议：HTTP
- 请求方式：默认情况下，数据请求和提交都用 POST(application/json)
- 返回数据格式：HTTP 状态为 200 时，返回 JSON 数据，否则为 HTTP 对应错误
- 登录认证方式：在 Cookie 中携带 sid=xxxxxxxxx
- 文件上传: 访问方式：POST(multipart/form-data)

二、返回 JSON 数据格式

格式如下，JSON 对象中的 status 属性为 API 状态码，为 0 时表示数据获取或操作成功，否则为相应的失败状态码。

```
{
  status: 0,
  enable: true,
  enable-web-control: true
  ...
}
```

API 状态码

```
{
  0: MW_STATUS_SUCCESS,
  1: MW_STATUS_PENDING,
  2: MW_STATUS_TIMEOUT,
  3: MW_STATUS_INTERRUPTED,
  4: MW_STATUS_TRY_AGAIN,
  5: MW_STATUS_NOT_IMPLEMENTED,
  6: MW_STATUS_UNKNOWN_ERROR,
  7: MW_STATUS_INVALID_ARG,
  8: MW_STATUS_NO_MEMORY,
  9: MW_STATUS_UNSUPPORTED,
  10: MW_STATUS_FILE_BUSY,
  11: MW_STATUS_DEVICE_BUSY,
  12: MW_STATUS_DEVICE_LOST,
  13: MW_STATUS_IO_FAILED,
  14: MW_STATUS_READ_FAILED,
  15: MW_STATUS_WRITE_FAILED,
  16: MW_STATUS_NOT_EXIST,
  17: MW_STATUS_TOO_MANY,
  18: MW_STATUS_TOO_LARGE,
  19: MW_STATUS_OVERFLOW,
  20: MW_STATUS_UNDERFLOW,
  21: MW_STATUS_FORMAT_ERROR,
  22: MW_STATUS_FILE_EXISTS,
  23: MW_STATUS_FILE_TYPE_ERROR,
  24: MW_STATUS_DEVICE_TYPE_ERROR,
  25: MW_STATUS_IS_DIRECTORY,
  26: MW_STATUS_READ_ONLY,
  27: MW_STATUS_RANGE_ERROR,
  28: MW_STATUS_BROKEN_PIPE,
  29: MW_STATUS_NO_SPACE,
  30: MW_STATUS_NOT_DIRECTORY,
  31: MW_STATUS_NOT_PERMITTED,
  32: MW_STATUS_BAD_ADDRESS,
  33: MW_STATUS_SEEK_ERROR,
  34: MW_STATUS_CROSS_DEVICE_LINK,
  35: MW_STATUS_NOT_INITIALIED,
  36: MW_STATUS_AUTH_FAILED,
  37: MW_STATUS_NOT_LOGGED_IN,
  38: MW_STATUS_WRONG_STATE,
  39: MW_STATUS_MISMATCH,
  40: MW_STATUS_VERIFY_FAILED,
  41: MW_STATUS_CONSTRAINT_VIOLATION
  42: MW_STATUS_CANCELED,
  43: MW_STATUS_IN_PROGRESS,
  44: MW_STATUS_CONN_REFUSED,
  45: MW_STATUS_CONN_RESET,
  46: MW_STATUS_ADDR_IN_USE,
  47: MW_STATUS_NO_RESPONSE,
  48: MW_STATUS_INFO_CHANGED,
  49: MW_STATUS_INVALID_DATA,
  50: MW_STATUS_NEED_MORE_DATA,
  51: MW_STATUS_NO_BUFFER,
  52: MW_STATUS_BUFFER_TOO_SMALL,
  53: MW_STATUS_BUFFER_IS_EMPTY,
  54: MW_STATUS_BUFFER_IS_FULL
}
```

DEMO: 命令行

在不同操作系统中，可以安装 wget 和 curl 两个工具，安装后可以在命令行中通过 wget 或 curl 命令来调用 API。

不同操作系统中，下边示例的 cookie 文件存放位置不同，请根据实际情况修改。(以下示例基于Linux。)

wget

1.登录并保存 cookies

```
wget --save-cookies=sid.txt --keep-session-cookies --header="Content-Type: application/json" --post-data='{"usernam
e":"Admin", "password": "c1c224b03cd9bc7b6a86d77f5dace40191766c485cd55dc48caf9ac873335d6f"}' http://192.168.66.1/api/
user/login -d -q -O -
```

2.获取用户列表

```
wget --load-cookies=sid.txt --keep-session-cookies --header="Content-Type: application/json" --post-data='' http://19
2.168.66.1/api/user/get-all -d -q -O -
```

3.新增用户

```
wget --load-cookies=sid.txt --keep-session-cookies --header="Content-Type: application/json" --post-data='{"usernam
e":"test","password":"9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08"}' http://192.168.66.1/api/use
r/add -d -q -O -
```

curl

1.登录并保存 cookies

```
curl --cookie-jar sid.txt http://192.168.66.1/api/user/login -X POST -H 'Content-Type: application/json' -d'{"usernam
e":"Admin", "password": "c1c224b03cd9bc7b6a86d77f5dace40191766c485cd55dc48caf9ac873335d6f"}'
```

2.获取用户列表

```
curl --cookie sid.txt http://192.168.66.1/api/user/get-all -X POST -H 'Content-Type: application/json' -d ''
```

3.新增用户

```
curl --cookie sid.txt http://192.168.66.1/api/user/add -X POST -H 'Content-Type: application/json' -d '{"username":"t
est","password":"9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08"}'
```

ping 接口

判断设备是否可以访问，无需登录。

在 `固件更新` 、 `重置设备` 、 `修改 IP 地址` 等操作完成后，设备需要重启，可以通过该接口判断设备是否已经重启完成。

请求方式

```
GET/POST /api/ping
```

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

reboot 接口

此接口用于重启设备，重启后需要重新登录，仅管理员有权限。

重启过程需要几分钟时间，可以使用 ping 接口 判断设备是否已经重启。

请求方式

```
GET/POST /api/reboot
```

返回数据

```
{
  "status": 0,
  "delay": 5,
  "estimate-sec": 15
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
delay	延时多少秒执行重启，单位 s。
estimate-sec	估算重启时间，单位 s。

factory-reset-permission 接口

通过该接口判断是否允许重置设备, 无需登录。

请求方式

```
GET/POST /api/factory-reset-permission
```

返回数据

```
{
  "status": 0
  "reset-enable": true
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
reset-enable	是否支持重置功能，有效值：true/false。

factory-reset 接口

恢复出厂设置。

请求方式

```
GET/POST /api/factory-reset
```

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/aoip/summary-info 接口

获取运行信息。

请求方式

```
POST /api/aoip/summary-info
```

返回数据

```
{
    "status": 0,
    "temperature": "79.61°C",
    "card-address": 0,
    "unbalance": {
        "in": {
            "linked": true,
            "depth": "L24",
            "channel-num": 2
        },
        "out": {
            "linked": false,
            "depth": "L24",
            "channel-num": 2
        }
    },
    "balance": {
        "in": {
            "linked": false,
            "depth": "L24",
            "channel-num": 2
        },
        "out": {
            "linked": false,
            "depth": "L24",
            "channel-num": 2
        }
    },
    "uac": {
        "usb-connected": true,
        "in": {
            "linked": false,
            "sample-rate": 48000,
            "depth": "L24",
            "channel-num": 4
        },
        "out": {
            "linked": false,
            "sample-rate": 48000,
            "depth": "L24",
            "channel-num": 4
        }
    }
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
temperature	温度。
card-address	拨码开关地址。
unbalance.in.linked	unbalance输入是否连接， true：连接， false：未连接
unbalance.in.depth	unbalance输入采样位宽 L16， L24
unbalance.in.channel-num	unbalance输入通道数
unbalance.out.linked	unbalance输出是否连接
unbalance.out.depth	unbalance输出采样位宽 L16， L24
unbalance.out.channel-num	unbalance输出通道数
balance.in.linked	balance输入是否连接
balance.in.depth	balance输入采样位宽 L16， L24
balance.in.channel-num	balance输入通道数
balance.out.linked	balance输出是否连接
balance.out.depth	balance输出采样位宽 L16， L24
balance.out.channel-num	balance输出通道数
uac.usb-connected	USB 是否已连接， true：连接， false：未连接
uac.in.linked	UAC输入是否工作 true/false
uac.in.sample-rate	UAC输入采样率
uac.in.depth	UAC 输入采样位宽
uac.in.channel-num	UAC输入通道数
uac.out.linked	UAC输出是否工作 true/false
uac.out.sample-rate	UAC输出采样率
uac.out.depth	UAC输出采样位宽
uac.out.channel-num	UAC输出通道数

/api/aoip/general-range 接口

通用设置参数描述

请求方式

```
POST /api/aoip/general-range
```

返回数据


```
{
    "tx-sample-rate": [
        "44100",
        "48000",
        "88200",
        "96000"
    ],
    "unbld-in": [
        "+12dBu",
        "+4dBu",
        "0dBu",
        "-2dBu",
        "0dBV",
        "-10dBV"
    ],
    "unbld-out": [
        "+12dBu",
        "+4dBu",
        "0dBu",
        "-2dBu",
        "0dBV",
        "-10dBV"
    ],
    "bld-in": [
        "+24dBu",
        "+18dBu",
        "+4dBu",
        "0dBu",
        "-2dBV",
        "0dBV",
        "-10dBV"
    ],
    "bld-out": [
        "+18dBu",
        "+4dBu",
        "0dBu",
        "-2dBV",
        "0dBV",
        "-10dBV"
    ],
    "igmp": [
        "Auto",
        "IGMPv2",
        "IGMPv3"
    ],
    "audio-pattern": [
        "off",
        "input",
        "output"
    ],
    "uac-num-channels": [
        "4",
        "2"
    ],
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/aoip/general-info 接口

获取通用设置信息

请求方式

```
POST /api/aoip/general-info
```

返回数据

```
{
    "status": 0,
    "device-lock": false,
    "tx-sample-rate": "48000",
    "unbld-in": "+12dBu",
    "unbld-out": "+12dBu",
    "bld-in": "+18dBu",
    "bld-out": "+18dBu",
    "igmp": "Auto",
    "micbias": true,
    "audio-pattern": "off",
    "uac-num-channels": 2
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
device.deivce-lock	锁定配置
tx-sample-rate	设备采样率
unbld-in	unbalanced 输入音量调整
unbld-out	unbalanced 输出音量调整
bld-in	balanced 输入音量调整
bld-out	balanced 输出音量调整
igmp	IGMP 版本, 有效值Auto, IGMPv2, IGMPv3
micbias	使能MIC 供电电压, 有效值true/false
audio-pattern	test tone

/api/aoip/general-apply 接口

通用设置

请求方式

```
POST /api/aoip/general-apply
```

属性	说明
tx-sample-rate	设备采样率
unbld-in	unbalanced 输入音量调整
unbld-out	unbalanced 输出音量调整
bld-in	balanced 输入音量调整
bld-out	balanced 输出音量调整
igmp	IGMP 版本, 有效值Auto, IGMPv2, IGMPv3
micbias	使能MIC 供电电压, 有效值true/false
audio-pattern	test tone
led-identify	有效值true/false

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/tx/live-info 接口

TX 流信息

请求方式

```
POST /api/tx/live-info
```

返回数据


```
{
    "stream-no-max": 2,
    "streaming-count-max": 16,
    "streaming-count": 3,
    "streaming": [
        {
            "uid": 3,
            "enable": true,
            "name": "RTSP Server",
            "stream-no": "Stream1",
            "type": "rtsp",
            "rtsp": {
                "bind-port": 554,
                "max-client-num": 8,
                "key": "aud",
                "enable-auth": false,
                "username": "",
                "password": ""
            },
            "aac-bitrate-kbps": "128",
            "report": {
                "clients-status": [
                    {
                        "audio-lost-percent": 5847.757058208435,
                        "audio-lost-total": 16777215,
                        "bitrate-kbps": 125,
                        "living-time-ms": 65359,
                        "name": "10.10.14.202",
                        "peer-audio-port": 63398,
                        "peer-rtsp-port": 56881,
                        "peer-video-port": 0,
                        "transport": "udp",
                        "video-lost-percent": 0.0,
                        "video-lost-total": 0
                    }
                ],
                "living-time-ms": 143812,
                "module-name": "mws_rtsp_sink_0",
                "module-type": 98,
                "num-clients": 1
            }
        },
        {
            "uid": 11,
            "enable": true,
            "name": "NDI",
            "stream-no": "Stream1",
            "type": "ndi",
            "ndi": {
                "source-name": "test11",
                "group-name": "public",
                "enable-full": true,
                "audio-standard": "SMPTE",
                "enable-discovery": false,
                "discovery-server": "",
                "transport-mode": "tcp-unicast",
                "mcast-addr": "",
                "mcast-mask": "",
                "mcast-ttl": 4,
```

```
            "enable-fail-over": false,
            "fail-over-ndi-name": "",
            "fail-over-ip-addr": "",
            "enable-web-control": true,
            "enable-logo": false
        },
        "aac-bitrate-kbps": "128",
        "report": {
            "module-name": "mws_ndi_sink_0",
            "module-type": 34,
            "ndi-name": "PRO-CONVERT-AES67 (test11)",
            "num-clients": 0
        }
    },
    {
        "uid": 9,
        "enable": false,
        "name": "TS over SRT",
        "stream-no": "Stream2",
        "type": "srt",
        "srt": {
            "mode": "listener",
            "dst-ip": "",
            "dst-port": 8000,
            "bind-port": 10000,
            "stream-id": "12/12",
            "connect-timeout": 3000,
            "retry-duration": 3000,
            "latency": 120,
            "bandwidth": 25,
            "mtu": 1500,
            "enc": "disable",
            "passphrase": "",
            "enable-logo": false
        },
        "aac-bitrate-kbps": "128",
        "report": {
            "mode": "listener",
            "module-name": "mws_srt_sink_0",
            "module-type": 114
        }
    }
],
"discovery": [
    {
        "is-ndi": true,
        "ndi-name": "DESKTOP-KN2V7CQ (Intel UHD Graphics 630 1)",
        "ndi-url": "192.168.65.2:5961"
    }
],
"status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
stream-no-max	stream 并发数
streaming-count-max	最大并发数量。
streaming-count	流数量。
streaming[i].uid	唯一ID, 须大于0。
streaming[i].enable	流使能, 有效值true/false。
streaming[i].name	流名称， 长度范围[1, 63]。
streaming[i].stream-no	stream选择， 有效值Stream1, Stream2。
streaming[i].type	流协议， 有效值rtsp, srt, ndi。
streaming[i].rtsp.bind-port	RTSP 端口。
streaming[i].rtsp.max-client-num	RTSP 最大连接数。
streaming[i].rtsp.key	RTSP 主码流key, 长度范围[1,63]。
streaming[i].rtsp.enable-auth	RTSP 使能Authentication, 有效值true/false。
streaming[i].rtsp.username	RTSP Authentication 用户名, 长度范围[0,63]。
streaming[i].rtsp.password	RTSP Authentication 密码, 长度范围[0,63]。
streaming[i].rtsp.enable-logo	显示LOGO, 有效值true/false
streaming[i].rtsp.report	RTSP 主码流report 信息。
streaming[i].srt.mode	TS over SRT 模式, 支持caller, listener。
streaming[i].srt.dst-ip	TS over caller 模式目的IP。
streaming[i].srt.dst-port	TS over SRT caller 模式目的, 有效值[1,65535]。
streaming[i].srt.bind-port	TS over SRT listener模式绑定端口, 有效值[1,65535]。
streaming[i].srt.stream-id	TS over SRT Stream ID, 长度范围[0,63]。
streaming[i].srt.connect-timeout	TS over SRT 连接超时，单位ms。
streaming[i].srt.retry-duration	TS over SRT 重试等待时间，单位ms。
streaming[i].srt.latency	TS over SRT 延时时间，单位ms。
streaming[i].srt.bandwidth	TS over SRT 带宽，百分比。
streaming[i].srt.mtu	TS over SRT MTU[228, 1500]。
streaming[i].srt.enc	TS over SRT 加密方式， 有效值disable，aes-128，aes-192，aes-256。
streaming[i].srt.passphrase	TS over SRT 秘钥, 长度范围[1,79]。
streaming[i].srt.enable-logo	显示LOGO, 有效值true/false。
streaming[i].ndi.source-name	视频源名称，默认为设备序列号
streaming[i].ndi.group-name	组名，默认为public
streaming[i].ndi.enable-full	使能NDI Full, 有效值true/false
streaming[i].ndi.audio-standard	音量标准，有效值SMPTE， EBU
streaming[i].ndi.enable-discovery	是否启用发现服务, 有效值true/false
streaming[i].ndi.discovery-server	发现服务器 IP 地址, 长度范围[1,63]
streaming[i].ndi.transport-mode	传输模式，支持udp-unicast，udp-multicast，rudp-unicast，tcp-unicast，tcp-multi
streaming[i].ndi.mcast-addr	组播地址

属性	说明
streaming[i].ndi.mcast-mask	组播掩码
streaming[i].ndi.mcast-ttl	TTL, 有效值[1,255]
streaming[i].ndi.enable-fail-over	是否启用备用通道, 有效值true/false
streaming[i].ndi.fail-over-ndi-name	备用通道视频源名称, 长度范围[1,63]
streaming[i].ndi.fail-over-ip-addr	备用通道 IP 地址, 长度范围[1,63]
streaming[i].ndi.enable-web-control	显示 Web 控制, 有效值true/false
streaming[i].ndi.enable-logo	显示LOGO, 有效值true/false
streaming[i].aac-bitrate-kbps	AAC码率, 支持128,192,256
discovery[i].is-ndi	自动发现的设置是否是NDI, 有效值true/false
discovery[i].ndi-name	自动发现NDI 名称
discovery[i].ndi-url	自动发现NDI 地址

/api/tx/live-apply 接口

TX 流设置

请求方式

```
POST /api/tx/live-apply
```

属性	说明
uid	唯一ID, 须大于0。
enable	流使能, 有效值true/false。
name	流名称， 长度范围[1, 63]。
stream-no	stream选择， 有效值Stream1, Stream2。
type	流协议， 有效值rtsp, srt, ndi。
rtsp.bind-port	RTSP 端口。
rtsp.max-client-num	RTSP 最大连接数。
rtsp.key	RTSP 主码流key, 长度范围[1,63]。
rtsp.enable-auth	RTSP 使能Authentication。
rtsp.username	RTSP Authentication 用户名, 长度范围[0,63]。
rtsp.password	RTSP Authentication 密码, 长度范围[0,63]。
rtsp.enable-logo	显示LOGO, 有效值true/false
rtsp.report	RTSP 主码流report 信息。
srt.mode	TS over SRT 模式, 支持caller, listener。
srt.dst-ip	TS over caller 模式目的IP。
srt.dst-port	TS over SRT caller 模式目的端口, 有效值[1,65535]。
srt.bind-port	TS over SRT listener模式绑定端口, 有效值[1,65535]。
srt.stream-id	TS over SRT Stream ID, 长度范围[0,63]。
srt.connect-timeout	TS over SRT 连接超时，单位ms。
srt.retry-duration	TS over SRT 重试等待时间，单位ms。
srt.latency	TS over SRT 延时时间，单位ms。
srt.bandwidth	TS over SRT 带宽，百分比。
srt.mtu	TS over SRT MTU[228, 1500]。
srt.enc	TS over SRT 加密方式， 有效值disable，aes-128，aes-192，aes-256。
srt.passphrase	TS over SRT 秘钥, 长度范围[1,79]。
srt.enable-logo	显示LOGO, 有效值true/false。
ndi.source-name	视频源名称，默认为设备序列号
ndi.group-name	组名，默认为public
ndi.enable-full	使能NDI Full, 有效值true/false
ndi.audio-standard	音量标准，有效值SMPTE， EBU
ndi.enable-discovery	是否启用发现服务, 有效值true/false
ndi.discovery-server	发现服务器 IP 地址
ndi.transport-mode	传输模式，支持udp-unicast，udp-multicast，rudp-unicast，tcp-unicast，tcp-multi
ndi.mcast-addr	组播地址
ndi.mcast-mask	组播掩码
ndi.mcast-ttl	TTL, 有效值[1,255]
ndi.enable-fail-over	启用备用通道, 有效值true/false
ndi.fail-over-ndi-name	备用通道视频源名称, 长度范围[1,63]

属性	说明
ndi.fail-over-ip-addr	备用通道 IP 地址, 长度范围[1,63]
ndi.enable-web-control	显示 Web 控制, 有效值true/false
ndi.enable-logo	显示LOGO, 有效值true/false
aac-bitrate-kbps	AAC码率, 支持128,192,256

e.g.

```
// SRT
{
    "uid": 9,
    "enable": true,
    "name": "TS over SRT",
    "type": "srt",
    "srt": {
        "select": 0,
        "mode": "listener",
        "dst-ip": "",
        "dst-port": 8000,
        "bind-port": 10000,
        "stream-id": "12/12",
        "connect-timeout": 3000,
        "retry-duration": 3000,
        "latency": 120,
        "bandwidth": 25,
        "mtu": 1500,
        "enc": "disable",
        "passphrase": ""
    },
    "aac-bitrate-kbps": "128"
}

// NDI
{
    "uid": 11,
    "enable": true,
    "name": "NDI",
    "type":"ndi",
    "ndi": {
        "source-name": "test11",
        "group-name": "public",
        "enable-discovery": false,
        "discovery-server": "",
        "transport-mode": "tcp-unicast",
        "mcast-ttl": 4,
        "mcast-addr": "",
        "mcast-mask": "",
        "enable-fail-over": false,
        "fail-over-ndi-name": "",
        "fail-over-ip-addr": "",
        "enable-web-control": true
    },
    "aac-bitrate-kbps": "128"
}
```

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/tx/live-del 接口

TX 流删除

请求方式

```
POST /api/tx/live-del
```

属性	说明
uid	唯一ID, 须大于0。

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/rx/live-info 接口

RX 流信息

请求方式

```
POST /api/rx/live-info
```

返回数据

```
{
    "stream-no-max": 2,
    "mw-buffer-duration-min": 1,
    "mw-buffer-duration-max": 3000,
    "mw-buffer-duration-def": 100,
    "gst-count-max": 16,
    "gst-count": 3,
    "gst": [
        {
            "uid": 9,
            "enable": false,
            "name": "TS over SRT",
            "stream-no": "Stream1",
            "url": "srt://10.10.11.54:10000?streamid=11&mode=caller&passphrase=111111111111&latency=100&mw-audio-trac
k=2&mw-buffer-duration=100&mw-headroom-db=0"
        },
        {
            "uid": 11,
            "enable": false,
            "name": "NDI",
            "stream-no": "Stream1",
            "url": "ntkndi://ndi?ndi-name=DESKTOP-KN2V7CQ (Intel UHD Graphics 630 1)&ndi-url=&mw-buffer-duration=100&
mw-headroom-db=0&mw-audio-standard=SMPTE"
        }
    ],
    "ndi-options": {
        "enable-discovery": false,
        "discovery-server": "",
        "group-name": "",
        "extra-ips": ""
    },
    "discovery": [],
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
stream-no-max	stream 并发数
mw-buffer-duration-min	最小缓存时间，单位ms。
mw-buffer-duration-max	最大缓存时间，单位ms。
mw-buffer-duration-def	默认缓存时间，单位ms。
gst-count-max	最大流数量。
gst-count	流数量。
gst[i].uid	唯一ID, 须大于0。
gst[i].enable	流使能true/false。
gst[i].name	流名称， 长度范围[1, 1023]。
gst[i].stream-no	stream选择， 有效值Stream1, Stream2。
gst[i].url	流链接。
gst[i].report	报告信息。
ndi-options[i].enable-discovery	启用discovery服务，有效值：true/false
ndi-options[i].discovery-server	服务器 IP 地址
ndi-options[i].group-name	客户端群组名称，默认 public
ndi-options[i].extra-ips	外部NDI source
discovery[i].is-ndi	自动发现是否是NDI， 有效值true/false
discovery[i].ndi-name	自动发现NDI 名称
discovery[i].ndi-url	自动发现NDI 地址

/api/rx/live-apply 接口

设置RX 流

请求方式

```
POST /api/rx/live-apply
```

属性	说明
uid	唯一ID, 须大于0。
enable	流使能true/false。
name	流名称， 长度范围[1, 63]。
stream-no	stream选择， 有效值Stream1, Stream2。
url	流链接。

1. SRT

```
// Caller 模式
srt://ip:port?mode=caller&streamid=12323&passphrase=12345678914&latency=123&mw-audio-track=1&mw-buffer-duration=100

// Listener 模式
srt://0.0.0.0:port?mode=listener&streamid=12323&passphrase=12345678914&latency=123&mw-audio-track=1&mw-buffer-duratio
n=100
```

URL 组成	说明
url	Listener：0.0.0.0 caller：合法 IP 地址（不能为：0.0.0.0）
port	端口号，有效范围：1 ~ 65535
mode	模式，有效值：caller/listener
streamid	streamid
latency	延迟时间,有效范围：20 ~ 8000
encryption	是否加密，有效值：true/false
passphrase	加密密码，选填，需要加密时才设置，长度：10 ~ 79
mw-audio-track	音轨，有效范围：1 ~ 8
mw-buffer-duration	缓冲时间（ms），有效值可通过 rx-live-info 接口获取
mw-headroom-db	headroom， 单位dB

2. NDI

```
ntkndi://ndi?ndi-name=DESKTOP-KN2V7CQ (Intel UHD Graphics 630 1)&ndi-url=&mw-buffer-duration=100&mw-headroom-db=0&&mw
-audio-standard=SMPTE
```

URL 组成	说明
ndi-name	NDI设备名称, 长度范围：0 ~ 127
ndi-url	NDI设备URL, 长度范围：0 ~ 127
mw-buffer-duration	缓冲时间（ms），有效值可通过 rx-live-info 接口获取
mw-headroom-db	headroom， 单位dB
mw-audio-standard	音量标准， 有效值SMPTE， EBU

返回数据

```
{
        "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/rx/live-del 接口

设置rx 流

请求方式

```
POST /api/rx/live-del
```

属性	说明
uid	唯一ID, 须大于0。

返回数据

```
{
        "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/rx/live-ndi-options-apply 接口

设置RX NDI 配置信息

请求方式

```
POST /api/rx/ndi-options-apply
```

属性	说明
enable-discovery	启用discovery服务，有效值：true/false
discovery-server	服务器 IP 地址
group-name	客户端群组名称，默认 public
extra-ips	外部NDI source

返回数据

```
{
        "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/aoip/matrix-desc 接口

矩阵结构描述

请求方式

```
POST /api/aoip/matrix-desc
```

返回数据


```
{
    "status": 0,
    "max-tx-channels": 32,
    "max-rx-channels": 32,
    "tx": [
        {
            "name": "Unbalance",
            "channel-count": 2,
            "channel-start": 0,
            "channel-end": 1,
            "show": true
        },
        {
            "name": "Balance",
            "channel-count": 2,
            "channel-start": 2,
            "channel-end": 3,
            "show": true
        },
        {
            "name": "UAC",
            "channel-count": 4,
            "channel-start": 4,
            "channel-end": 7,
            "show": true
        },
        {
            "name": "Dante",
            "channel-count": 8,
            "channel-start": 8,
            "channel-end": 15,
            "show": true
        },
        {
            "name": "Stream",
            "channel-count": 4,
            "channel-start": 16,
            "channel-end": 19,
            "show": true
        },
        {
            "name": "Stream",
            "channel-count": 4,
            "channel-start": 20,
            "channel-end": 23,
            "show": true
        },
        {
            "name": "Unused",
            "channel-count": 4,
            "channel-start": 24,
            "channel-end": 27,
            "show": false
        },
        {
            "name": "Unused",
            "channel-count": 4,
            "channel-start": 28,
            "channel-end": 31,
```

```
            "show": false
        }
    ],
    "rx": [
        {
            "name": "Unbalance",
            "channel-count": 2,
            "channel-start": 0,
            "channel-end": 1,
            "show": true
        },
        {
            "name": "Balance",
            "channel-count": 2,
            "channel-start": 2,
            "channel-end": 3,
            "show": true
        },
        {
            "name": "UAC",
            "channel-count": 4,
            "channel-start": 4,
            "channel-end": 7,
            "show": true
        },
        {
            "name": "Dante",
            "channel-count": 8,
            "channel-start": 8,
            "channel-end": 15,
            "show": true
        },
        {
            "name": "Stream",
            "channel-count": 4,
            "channel-start": 16,
            "channel-end": 19,
            "show": true
        },
        {
            "name": "Stream",
            "channel-count": 4,
            "channel-start": 20,
            "channel-end": 23,
            "show": true
        },
        {
            "name": "Unused",
            "channel-count": 4,
            "channel-start": 24,
            "channel-end": 27,
            "show": false
        },
        {
            "name": "Unused",
            "channel-count": 4,
            "channel-start": 28,
            "channel-end": 31,
            "show": false
        }
```

```
    ]
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
max-tx-channels	设备支持最大的TX 通道数
max-rx-channels	设备支持最大的RX 通道数
tx[i].name	组通道名称
tx[i].channel-count	组通道数
tx[i].channel-start	组通道开始
tx[i].channel-start	组通道结束
tx[i].show	组是否隐藏，有效值true/fasle
rx[i].name	组通道名称
rx[i].channel-count	组通道数
rx[i].channel-start	组通道开始
rx[i].channel-start	组通道结束
rx[i].show	组是否隐藏，有效值true/fasle

/api/aoip/matrix-info 接口

矩阵信息

请求方式

```
POST /api/aoip/matrix-info
```

返回数据


```
{
    "status": 0,
    "max-tx-channels": 22,
    "max-rx-channels": 46,
    "matrix": [
        {
            "tx-no": 0,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 1,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 2,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 3,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [
```

```
            ]
        },
        {
            "tx-no": 4,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 5,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 6,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 7,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 8,
            "tx-mute": false,
            "tx-volume": 0,
```

```
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 9,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 10,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 11,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 12,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
```

```
            "rx-volume": [

            ]
        },
        {
            "tx-no": 13,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 14,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 15,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 16,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 17,
```

```
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 18,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 19,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 20,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [

            ],
            "rx-volume": [

            ]
        },
        {
            "tx-no": 21,
            "tx-mute": false,
            "tx-volume": 0,
            "mix-state": [

            ],
            "rx-mute": [
```

```
        ],
        "rx-volume": [

        ]
    }
],
"uac-rx": {
    "channel-num": 4
},
"stream-rx": [
    {
        "stream-no": "Stream1",
        "stream-index": 4,
        "name": "NDI",
        "type": "ndi",
        "channel-num": 4
    }
],
    "uac-tx": {
        "channel-num": 4
    },
    "stream-tx": [
        {
            "stream-no": "Stream1",
            "stream-index": 4,
            "name": "ndi-01",
            "type": "ndi",
            "channel-num": 4
        }
    ]
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
max-tx-channels	设备支持最大的TX 通道数
max-rx-channels	设备支持最大的RX 通道数
matrix[i].tx-no	TX 通道编号
matrix[i].tx-mute	TX 静音 值false/true
matrix[i].tx-volume	TX 端音量 [-36, 36]， 单位dB
matrix[i].mix-state	TX-RX连接状态: 0: 未连接 1： 连接成功 2:连接中 3：连接失败
matrix[i].rx-mute	TX-RX 静音 值false/true
matrix[i].rx-volume	TX-RX音量 [-36, 36]， 单位dB
uac-rx.channel-num	UAC RX 通道数
live-rx[i].stream-no	流选择
live-rx[i].stream-index	流索引
live-rx[i].name	流名称
live-rx[i].type	流协议， 有效值srt, ndi
live-rx[i].channel-num	流通道数
uac-tx.channel-num	UAC TX 通道数
live-tx[i].stream-no	流选择
live-tx[i].stream-index	流索引
live-tx[i].name	流名称
live-tx[i].type	流协议， 有效值rtsp, srt, ndi
live-tx[i].channel-num	流通道数

/api/aoip/audio-meter 接口

矩阵全部音量信息

请求方式

```
POST /api/aoip/audio-meter
```

返回数据


```
{
    "tx-db": [
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -81,
        -81,
        -81,
        -81,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100
    ],
    "rx-db": [
        -88,
        -83,
        -95,
        -91,
        -100,
        -100,
        -100,
        -100,
        -26,
        -26,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
```

```
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100,
        -100
    ],
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
tx-db	dBFs值
rx-db	dBFs值

/api/aoip/audio-meter-limit 接口

获取矩阵指定音量信息

请求方式

```
POST /api/aoip/audio-meter-limit
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
tx-no	TX 通道号
rx-no	RX 通道号

e.g.

```
{
    "tx-no": 1,
    "rx-no": [1, 2, 3, 4]
}
```

返回数据

```
{
    "tx-dBFS": -100,
    "tx-min-db": -100,
    "tx-max-db": 40,
    "rx-channels": [
        {
            "rx-no": 1,
            "dBFS": -87,
            "min-db": -100,
            "max-db": 40
        },
        {
            "rx-no": 2,
            "dBFS": -91,
            "min-db": -100,
            "max-db": 40
        },
        {
            "rx-no": 3,
            "dBFS": -88,
            "min-db": -100,
            "max-db": 40
        },
        {
            "rx-no": 4,
            "dBFS": -100,
            "min-db": -100,
            "max-db": 40
        }
    ],
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
tx-dBFS	TX通道dBFs值
tx-min-db	TX通道dBFs最小值
tx-min-db	TX通道dBFs最大值
rx-channels[i].rx_no	RX 通道号
rx-channels[i].dBFS	RX通道dBFs值
rx-channels[i].min-db	RX通道号dBFs最小值
rx-channels[i].max-db	RX通道号dBFs最大值

/api/aoip/matrix-settings 接口

设置矩阵TX-RX连接

请求方式

```
POST /api/aoip/matrix-settings
```

属性	说明
matrix[i].tx-no	TX 通道编号
matrix[i].rx-no	RX 通道编号
matrix[i].mix	TX-RX连接状态 , 0: 未连接， 1： 连接

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/aoip/matrix-clear 接口

清空矩阵设置

请求方式

```
POST /api/aoip/matrix-clear
```

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/aoip/volume-tx 接口

设置矩阵TX音量

请求方式

```
POST /api/aoip/volume-tx
```

属性	说明
volumes[i].tx-no	TX 通道号
volumes[i].tx-mute	TX 静音 值false/true
volumes[i].db	音量 [-36, 36]， 单位dB

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/aoip/volume-mix 接口

设置矩阵TX-RX音量

请求方式

```
POST /api/aoip/volume-mix
```

属性	说明
volumes[i].tx-no	TX 通道号
volumes[i].rx-no	RX 通道号
volumes[i].rx-mute	静音
volumes[i].db	音量 [-36, 36]， 单位dB

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/dante/state 接口

dante 运行状态

请求方式

```
POST /api/dante/state
```

返回数据

```
{
    "device": {
        "deivce-lock": false
    },
    "clock-basic": {
        "uuid": "FE:30:38:71:E3:C7",
        "master-uuid": "00:1D:C1:50:B6:D8",
        "grandmaster-uuid": "00:1D:C1:50:B6:D8",
        "is-mute": false,
        "is-locked": true,
        "freq-ppm": 2
    },
    "clock-params": {
        "priority1": 254,
        "priority2": 116,
        "domain": 0,
        "sync-interval": 0,
        "announce-interval": 0,
        "ttl": 16
    },
    "audio-format": {
        "sample-rate": 48000,
        "encoding": "PCM24"
    },
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
device.deivce-lock	dante 锁定配置
clock-basic.uuid	UUID
clock-basic.master-uuid	master UUID
clock-basic.grandmaster-uuid	grandmaster UUID
clock-basic.is-mute	是否静音
clock-basic.is-locked	始终是否锁定
clock-basic.freq-ppm	频率PPM
clock-params.priority1	PTP优先级1
clock-params.priority2	PTP优先级2
clock-params.domain	PTP domain
clock-params.sync-interval	sync消息间隔
clock-params.announce-interval	announce消息间隔
audio-format.sample-rate	采样率
audio-format.encoding	音频编码

/api/aoip/export-reports 接口

导出dante日志

请求方式

```
POST /api/aoip/export-reports
```

/system/device-info 接口

获取设备信息。请注意capability各子项是否为true，当true时相应的API访问才有效

请求方式

```
POST /api/system/device-info
```

返回数据

```
{
    "device-name": "Pro Router ONE",
    "product-id": "0x601",
    "product-name": "Pro Router ONE",
    "hardware-rev": "B",
    "serial-number": "0123456789",
    "firmware-ver": "0.9.210",
    "firmware-name": "Development",
    "build-time": "2023-04-14 06:48:13",
    "capability": {
        "support-usbc-name": false,
        "support-timezone": true,
        "support-ntp": true,
        "support-station": true,
        "support-ap": true,
        "support-online-upgrade": false,
        "support-sc-control": false,
        "support-if-prio": false,
        "support-usb-ncm": false,
        "support-wifi-mutex": false,
        "support-ipv6": true
    },
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
product-id	设备id。
product-name	设备名称。
hardware-rev	硬件版本。
serial-number	设备序列号。
firmware-ver	设备固件版本。
firmware-name	设备固件名称。
build-time	设备固件构建时间。
capability.support-timezone	支持时区。
capability.support-ntp	支持NTP。
capability.support-station	WIFI 支持STA模式。
capability.support-ap	WIFI 支持AP模式。
capability.support-online-upgrade	支持在线升级。
capability.support-sc-control	支持云管理。
capability.support-sc-control	支持云管理。
capability.support-if-prio	支持interface优先级控制。
capability.support-usb-ncm	支持关闭ncm, reboot生效。
capability.support-wifi-mutex	WIFI工作模式互斥。
capability.support-ipv6	支持IPv6。

/system/info 接口

获取cpu， 内存信息等信息。

请求方式

```
POST /api/system/info
```

返回数据

```
{
    "device-name": "USB Fusion",
    "uptime": 8410,
    "cpu": {
        "total": 1624896,
        "idle": 1281701,
        "usage": 2110
    },
    "mem": {
        "total": 8069612,
        "avail": 7171768
    },
    "datetime": {
        "cur-time": "2021-12-20 13:25:57",
        "zonename": "Asia/Shanghai",
        "ntp-enable": true,
        "ntp-server1": "0.pool.ntp.org",
        "ntp-server2": "1.pool.ntp.org"
    },
  "auto-reboot":{
    "enable":true,
    "hour":23,
    "min":59,
    "week":[
      1,
      2
    ]
  },
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
device-name	设备名称。
uptime	开机时长， 单位s。
cpu.total	CPU 总时间。
cpu.idle	CPU 空闲时间。
cpu.usage	CPU 使用率 x 100。
mem.total	系统总内存， 单位KB。
mem.avail	系统可用内存，单位KB。
datetime.cur-time	系统时间，格式：yyyy-MM-dd HH:mm:ss。
datetime.zonename	时区名称。
datetime.ntp-enable	NTP 使能。
datetime.ntp-server1	NTP服务器1。
datetime.ntp-server2	NTP服务器2。
auto-reboot.enable	使能auto-reboot
auto-reboot.hour	时， 24小时制
auto-reboot.min	分
auto-reboot.week	周

/system/set-device-name 接口

设置设备名称。

请求方式

```
POST /api/system/set-device-name
```

参数	说明
name	设备名称。

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/system/set-date-time 接口

NTP 功能设置。

请求方式

```
POST /api/system/set-date-time
```

参数	说明
ntp-enable	是否使能NTP。
ntp-server1	NTP 服务器1。
ntp-server2	NTP 服务器2。
time	本地时间，格式：yyyy-MM-dd HH:mm:ss。

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/system/timezone-set 接口

时区设置。

请求方式

```
POST /api/system/timezone-set
```

参数	说明
zonename	时区名称。

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/system/auto-reboot 接口

定时重启。

请求方式

```
POST /api/auto-reboot
```

参数	说明
auto-reboot.enable	使能auto-reboot
auto-reboot.hour	时， 24小时制
auto-reboot.min	分
auto-reboot.week	周

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/api/aoip/export-settings 接口

导出配置

请求方式

```
POST /api/aoip/export-settings
```

/api/aoip/import-settings 接口

导入配置

请求方式

```
POST /api/aoip/import-settings
```

/network/if-info 接口

网卡信息。

请求方式

```
POST /api/network/if-info
```

返回数据


```
{
    "device-name": "USB Fusion yxy1",
    "net": [
        {
            "enable": true,
            "iface": "eth0",
            "type": 0,
            "use-dhcp": true,
            "ipaddr": "10.10.12.166",
            "netmask": "255.255.240.0",
            "gateway": "10.10.0.1",
            "mac": "84:85:86:87:88:2e",
            "link-speed": 1000,
            "link-state": 2,
            "tx-speed-kbps": 0,
            "rx-speed-kbps": 107
        },
        {
            "enable": true,
            "iface": "wlan0",
            "type": 1,
            "mode": 1,
            "ssid": "USB-Fusion_yx_5G",
            "use-dhcp": true,
            "ipaddr": "192.168.67.1",
            "netmask": "255.255.255.0",
            "gateway": "",
            "mac": "10:2c:6b:fd:9b:78",
            "link-speed": -1,
            "link-state": 2,
            "tx-speed-kbps": 3,
            "rx-speed-kbps": 0
        },
        {
            "enable": true,
            "iface": "usb0",
            "type": 3,
            "use-dhcp": true,
            "ipaddr": "192.168.66.1",
            "netmask": "255.255.255.0",
            "gateway": "192.168.66.1",
            "mac": "8e:40:df:be:7c:fa",
            "link-speed": 480,
            "link-state": 2,
            "tx-speed-kbps": 0,
            "rx-speed-kbps": 0
        }
    ],
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
device-name	设备名称。
net[i].enable	网卡服务是否开启。
net[i].iface	网卡名称。
net[i].type	网卡类型 0: 以太网 1：WIFI 2: 4G 模块 3：USB。
net[i].mode	WIFI 工作模式, 当net[i].type == 1 存在， 0： STA 模式 1：AP 模式。
net[i].ssid	WIFI ssid。
net[i].reboot-require	WIFI 重启生效。
net[i].use-dhcp	true： 使用dhcp获取ip false： static 配置网络。
net[i].ipaddr	IP 地址。
net[i].netmask	子网掩码。
net[i].ipv6addr	IPv6 地址信息。
net[i].gateway	网关地址。
net[i].mac	MAC 地址。
net[i].link-speed	速率 10 : 10Mbps, 100: 100Mbps, 1000: 1Gbps, 2500: 2.5Gbps, 10000: 10Gbps。 usb 支持的速率 12: full-speed, 480: high-speed, 5000: super-speed-5g, 10000: super-speed-10g
net[i].link-state	端口状态 0： down, 1： disconnected, 2：connnected
net[i].vendor	4G模块厂商信息。
net[i].product	4G模块型号信息。
net[i].tx-speed-kbps	发送速度 (Kbps)。
net[i]. rx-speed-kbps	接收速度 (Kbps)。

/network/if-set 接口

配置网卡。

请求方式

```
POST /api/network/if-set
```

参数	说明
iface	网卡名称。
use-dhcp	true：使用DHCP获取IP；false：static 配置网络
ipaddr	IP 地址，当 use-dhcp 为 false 需带上
netmask	子网掩码， 当 use-dhcp 为 false 需带上
gateway	网关地址， 当 use-dhcp 为 false 需带上

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/network/if-route 接口

获取默认路由。

请求方式

```
POST /api/network/if-route
```

返回数据

```
{
    "iface": "",
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
ifname	默认路由经过的网卡，iface 若为空， 表示暂时无路由。

/network/get-dns 接口

获取DNS。

请求方式

```
POST /api/network/get-dns
```

返回数据

```
{
    "is-manual": false,
    "dns1": "10.0.1.3",
    "dns2": "",
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
is-manual	是否手动设置DNS。
dns1	DNS， 空字符表示未设置。
dns2	DNS， 空字符表示未设置。

/network/set-dns 接口

设置DNS。

请求方式

```
POST /api/network/set-dns
```

参数	说明
is-manual	是否手动设置DNS。
dns1	DNS， 空字符表示未设置。
dns2	DNS， 空字符表示未设置。

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/network/usb-config 接口

配置USB网卡。

请求方式

```
POST /api/network/usb-config
```

参数	说明
iface	网卡名称。
ipaddr	IP 地址。

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/certificate/info 接口

证书信息。

请求方式

```
POST /api/certificate/info
```

返回数据


```
{
    "status": 0,
    "enable": true,
    "certificate": {
        "subject-name": "/C=CN/ST=GuangDong/L=ShenZhen/O=bolin-av.com/CN=bolin-av.com/emailAddress=av@bolin-av.com",
        "issuer-name": "/C=CN/ST=GuangDong/L=ShenZhen/O=Bolin Technology Root CA",
        "version": 1,
        "serial-num": "C11B50824C53B27C",
        "expires": "Mar 25 02:06:25 3022 GMT",
        "valid-before": "Nov 22 02:06:25 2022 GMT",
        "public-key": "30818902818100ba1e3aff73f880f3bc219f3d6714edb2cf4a4b8fb072cdf55c5058903af7691eeeb4cae6aacb7148
6b6cb2001e14cb5b9113d52f8db666c87b65465a4a1204976390d33ad42de91597bcab511d6ca9c0b7e3dad4f75844206721024066 05eb4a1dcbf
9871f85ec412947b27648ee48b03d2af9e8b9f915f534bec99d4d6ed3d70203010001",
        "algorithm": "1.2.840.113549.1.1.11",
        "type": "EVP_PKEY_RSA"
    }
}
```


属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/certificate/enable 接口

使能https访问， reboot后生效。

请求方式

```
POST /api/certificate/enable
```

参数	说明
enable	有效值true/false。

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/certificate/import 接口

导入SSL证书。

请求方式

```
POST /api/certificate/import
```

参数	说明
certificate	根证书。
private-key	根证书私钥。

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/certificate/delete 接口

删除SSL证书。

请求方式

```
POST /api/certificate/delete
```

返回数据

```
{
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/user/login 接口

用户登录，登录成功后会在 Cookie 存放 Session ID (Cookie: sid-[serial number]=t2i704wbvoy51y408p588bpji010ibp0)。

请求方式

```
POST /api/user/login
```

参数	说明
username	用户名。
password	密码，使用 sha256加密。

返回数据

```
{
    "status": 0,
    "sid": "t2i704wbvoy51y408p588bpji010ibp0"
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

接口示例

```
// login (username: Admin, password=Admin)
curl --cookie-jar sid.txt http://192.168.66.1/api/user/login -X POST -H 'Content-Type: application/json' -d'{"usernam
e":"Admin", "password": "c1c224b03cd9bc7b6a86d77f5dace40191766c485cd55dc48caf9ac873335d6f"}'
```

/user/logout 接口

退出登录，返回到登录界面。

请求方式

```
POST /api/user/logout
```

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/user/get-all 接口

获取系统用户列表信息，仅管理员有权限。

请求方式

```
POST /api/user/get-all
```

返回数据

```
{
    "users": [
        {
            "username": "Admin",
            "group": "Admin"
        }
    ],
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
users	用户列表数组。username：用户名，group：用户组。

/user/add 接口

添加用户，仅管理员有权限。

请求方式

```
POST /api/user/add
```

参数	说明
username	用户名。
password	密码，使用 sha256加密。

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/user/del 接口

删除用户，仅管理员有权限。

请求方式

```
POST /api/user/del
```

参数	说明
username	用户登录名。

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/user/ch-password 接口

用户修改自己的登录密码，修改时必须输入原密码。

请求方式

```
POST /api/user/ch-password
```

参数	说明
password	原密码，使用 sha256加密。
new-password	新密码，使用 sha256加密。

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/user/set-password 接口

重置用户密码，无需输入原密码，仅管理员有权限。

请求方式

```
POST /api/user/set-password
```

参数	说明
username	用户登录名。
password	新密码，使用 sha256加密。

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

upgrade/online-check 接口

启动在线升级检查。

请求方式

```
POST /api/upgrade/online-check
```

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

upgrade/online-check-result 接口

获取在线检查结果。

请求方式

```
POST /api/upgrade/online-check-result
```

返回数据

```
{
    "up-to-date": true,
    "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
up-to-date	当前版本是否为最新，有效值：true/false。
version	最新版本。
size	最新版本的大小。
md5	最新版本的md5值。
changeLog	最新版本升级内容

/upgrade/upload-fw 接口

上传固件，上传文件格式必须为.mwf, 必须使用POST multipart/form-data上传文件。

请求方式

```
POST /upgrade/upload-fw
```

返回数据

```
{
  "status": 0,
  "up-to-date": true,
  "version": "1.1.72"
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
up-to-date	是否是最新版。
version	上传的固件版本号。

/upgrade/update 接口

执行更新操作，更新过程中可以通过 /upgrade/state 接口获取当前状态。

请求方式

```
POST /api/upgrade/update
```

参数	说明
is-online	false：离线升级， true：在线升级
mode	升级模式选择 0：Auto 模式，自动选择 Upgrade/Factory/FactoryClear 模式 1：Upgrade 模式 2：Factory 模式 3：FactoryClear 模式
timeout	升级进度不变超时时间，单位 s。

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/upgrade/state 接口

获取当前固件版本信息和升级状态，仅管理员有权限。

请求方式

```
POST /api/upgrade/state
```

返回数据

```
{
  "status": 0,
  "state": "updating",
  "cur-ver": "1.1.72",
  "update-version": "1.1.72",
  "num-steps": 4,
  "step": 2,
  "step-name": "Erasing image",
  "step-progress": 28
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
state	任务执行状态，0：空闲，1：初始化升级，2：升级中，3：升级完成, 4: online 固件下载中
cur-ver	当前固件版本号
update-version	最新固件版本号
step	执行到第几步，仅在 state 为 2 状态下存在。
num-steps	总计需要几个步骤，仅在 state 为 2 状态下存在。
step-name	当前执行步骤的名称，仅在 state 为 2 状态下存在。
step-progress	当前执行的步骤的进度，值为 0 ~ 100，单位%, 仅在 state 为 2 状态下存在。
download-percent	在线下载百分比

/upgrade/clear 接口

清除升级状态。

请求方式

```
POST /upgrade/clear
```

返回数据

```
{
  "status": 0
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/log/clear 接口

清除全部系统日志，仅管理员有权限。

请求方式

```
POST /api/log/clear
```

返回数据

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。

/log/filter 接口

过滤日志。

请求方式

```
POST /api/log/filter
```

参数	说明
types	日志类型，有效值：all、info、warn、error，当有多个类型时用英文逗号隔开。
key	过滤关键字， 可以为空字符串。

返回数据

```
{
    "status": 0,
    "logs": [
        {
            "no": 0,
            "time": "2022/09/09 16:11:07.920",
            "type": "info",
            "message": "xxxxxx"
        },
        {
            "no": 1,
            "time": "2022/09/09 16:11:04.721",
            "type": "info",
            "message": "xxxxxx"
        }
    ]
}
```

属性	说明
status	0：执行成功。返回其它值请参考 API 状态码。
logs[i].no	编号。
logs[i].time	日期时间。
logs[i].type	输出级别，包括 info, warn, error。
logs[i].message	日志内容。

/log/export 接口

导出设备当前的系统日志，导出文件为 html 格式，仅管理员有权限。

请求方式

```
POST /api/log/export
```

参数	说明
filename	导出的文件名称

请求结果

直接下载 html 格式日志文件到本地。