# Senthink X-子设备接入

# 协议文档

# （v2.0）

杭州贤芯科技有限公司

2022 年 10 月 26 日

## 文档管理信息

| | |
|---|---|
| **文件名称** | Senthink X-子设备接入协议文档 |
| **文件编号** | |
| **协议版本号** | V2.0 |
| **保密级别** | 开放（ ） 机密（√） 绝密（ ） |
| **保存期限** | 短期（ ） 长期（√） 永久（ ） |
| **编制人员** | 张骁健、姚亚男 |

# 文档变更记录

| 协议版本 | 变更日期 | 变更内容 | 变更人 | 对应平台版本 |
|---|---|---|---|---|
| V1.0.0 | 2021-11-22 | 创建协议文档 | 张骁健 | V1.4 |
| V1.1.0 | 2022-01-26 | 调整子设备物模型服务调用(json)参数格式。文档描述“网关删除子设备”修改 | 张骁健、姚亚男 | V1.4 |

| | | | | |
|---|---|---|---|---|
| | | 为“网关删除子设备拓扑关系”。网关主动查询拓扑关系，平台应答报文中是否在线增加-1 未激活状态。变更子设备物模型 topic。修改子设备上线、物模型上报平台应答报文格式。修改子设备上线，code 定义。修改物模型应答报文格式。修改子设备上线协议，子设备上线前，平台上该网关与子设备关系必须为“已关联”。修改平台添加子设备协议下发数据格式，ClientID 改为 SubdeviceID，标题描述改为“平台通知网关添加子设备（白名单）”。修改网关主动查询拓扑关系，返回格式，增加 SubdeviceID 的返回，增加关联状态返回：未关联、已关联、解除中。 | | |
| V1.1.1 | 2022-06-01 | 服务调用 ACK 添加"ServiceIdentify"字段；子设备原有加密字段 Mode，现在统一改为只有不加密，保留字段；调整之前协议中，不规范的标点符号，多余空格； | 姚亚男 | V1.5 |
| V1.1.2 | 2022-07-25 | 添加拓扑关系，返回码“2”，新增免注册设备号校验错误； | 姚亚男 | V2.0 |
| V2.0 | 2022-10-26 | 协议优化，属性设置设备 ack 回复，物模型历史数据,修复 topic 列表物模型相关话题 | 姚亚男 | V2.2 |

## 版权声明



## 免责声明



# 1 网关与子设备

## 1.1 网关与子设备概述

在很多物联网场景中，终端设备本身没有连接互联网能力，那么数据如何上云呢?

IoT 物联网平台支持设备 MQTT 直连，也支持的设备挂载到网关上，作为网关的子设备，由网关代理接入 IoT 物联网平台。

这时候网关设备除了自身作为 IoT 网关设备与 IoT 物联网平台建立 MQTT 连接，收发数据，还要负责子设备的管理，包括:

1、子设备动态注册

2、获取云端网关下子设备列表

3、添加子设备、删除子设备

4、子设备上线、子设备下线

5、监听子设备禁用和删除

6、代理子设备上下行的能力

网关和子设备通信的协议暂使用 mqtt，逻辑由网关实现。

网关与子设备的拓扑关系如下图所示：

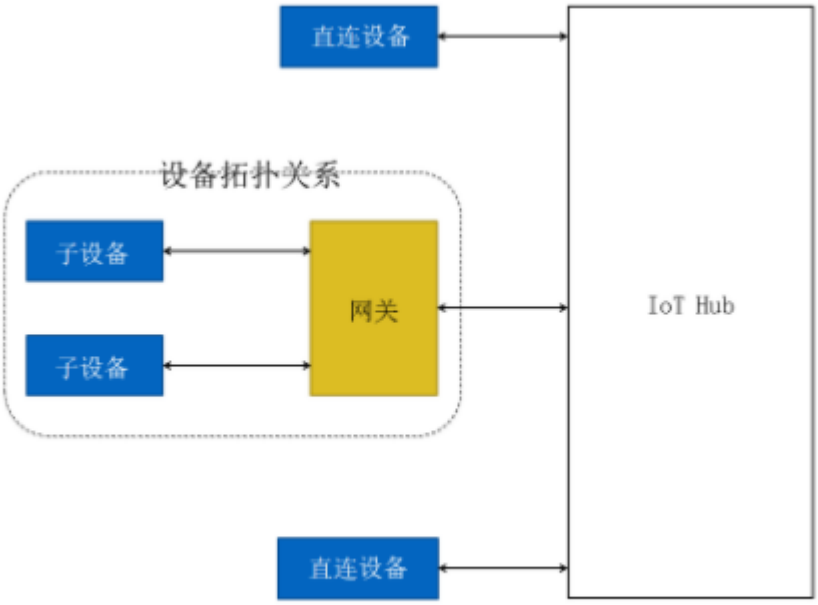



图 1.1.1 IoT 网关与子设备的拓扑图

## 1.2 子设备接入前准备事项

子设备在接入 SenthinkX 平台之前，需要先完成下事项：

1. 在平台提供的管理后台注册一个硬件厂商账号，云平台会为该账号分配一个厂商的唯一标识 OpenID；
2. 注册厂商账号后，需要在平台提供的管理后台上至少创建一个产品类，云平台会为每个产品分配一个产品的唯一标识 ProductID 以及产品的密钥 ProductKey（用于入网加密以及会话 Key 的更新，注意保密！）；
3. 网关接入 SenthinkX 平台之前，须要将基础参数烧录进设备。后面网关与平台的通信需

要这些参数才能完成，不同设备所需要烧录的参数如下：

（1） 一型一密免注册设备：GatewayDeviceID、OpenID、ProductID、ProductKey

（2） 一型一密预注册设备：GatewayDeviceID、OpenID、ProductID、ProductKey

（3） 一机一密预注册设备：GatewayDeviceID、OpenID、ProductID、ProductKey、ClientID、DeviceSecret

4. 子设备对接 SenthinkX 平台之前，也必须将基础参数烧录进子设备。网关认证完平台之后需要主动上报子设备的这些参数进行校验和上线，不同子设备所需要的参数如下：

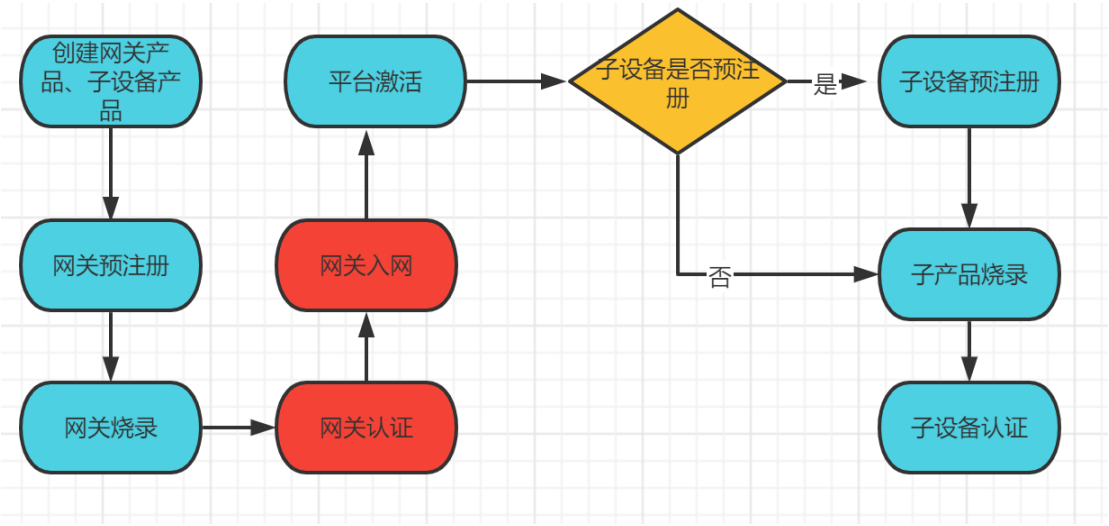
（4） 一型一密免注册设备：SubdeviceID、OpenID、ProductID、ProductKey

（5） 一型一密预注册设备：SubdeviceID、OpenID、ProductID、ProductKey

（6） 一机一密预注册设备：SubdeviceID、OpenID、ProductID、ProductKey、ClientID、DeviceSecret

至此，接入前的准备工作完成！

## 1.3 子设备接入流程




### 1.3.1 创建网关产品、子设备产品

在物联网平台控制台，分别创建网关产品和子设备产品。

网关产品的节点类型需选择为网关设备，子设备产品类型选择为网关子设备。产品创建后，

会生成对应的 OpenID、ProductID、ProductKey 信息（下面统称为产品证书），产品证书信息后续会用户进行账号密码生成，从而进行设备认证和设备入网。

### 1.3.2 网关预注册

网关预注册为一型一密预注册、一机一密预注册设备才有的流程，免注册网关直接跳过此流程即可。

对于一型一密预注册、一机一密预注册设备，在完成产品创建后网关参数烧录之前，需要将网关预先注册到平台中，平台为保证设备连接信息的安全性，为每一台设备都生成了唯一的设备密钥 DeviceSecret 和设备唯一标识 ClientID（非 GatewayDeviceID）。完成预注册的设备信息此时已录入平台，设备处于离线待激活状态，等待入网后自动激活。

### 1.3.3 网关烧录

网关接入 SenthinkX 平台之前，必须要将基础参数烧录进网关。后面网关与平台的通信需要这些参数才能完成，不同设备所需要烧录的参数如下：

（1） 一型一密免注册设备：GatewayDeviceID、OpenID、ProductID、ProductKey

（2） 一型一密预注册设备：GatewayDeviceID、OpenID、ProductID、ProductKey

（3） 一机一密预注册设备：GatewayDeviceID、OpenID、ProductID、ProductKey、ClientID、DeviceSecret

### 1.3.4 网关认证、激活

网关在能代理子设备进行通信之前，需要按照《SenthinkX-MQTT 智能设备接入协议文档》对网关设备进行认证。完成认证连接后，平台会将设备的状态标记为激活。详细操作步骤参考《SenthinkX-MQTT 智能设备接入协议文档》。

### 1.3.5 子设备预注册

子设备预注册为一型一密预注册、一机一密预注册设备才有的流程，免注册子设备直接跳过此流程。

对于一型一密预注册、一机一密预注册设备，在产品创建完成后子设备参数烧录之前，需要将子设备预先注册到平台中，平台为保证子设备连接信息的合法性，也为每一台子设备生成唯一的设备密钥 DeviceSecret、和设备唯一标识 ClientID（非 SubdeviceID）。

## 1.3.6 子设备产线烧录

子设备接入 SenthinkX 平台之前，必须要将基础参数烧录进设备。后面平台处理这些设备的数据需要这些参数验证通过才能完成，不同设备所需要烧录的参数如下：

（1）一型一密免注册设备：SubdeviceID、OpenID、ProductID、ProductKey

（2）一型一密预注册设备：SubdeviceID、OpenID、ProductID、ProductKey

（3）一机一密预注册设备：SubdeviceID、OpenID、ProductID、ProductKey、ClientID、DeviceSecret

## 1.3.7 子设备认证

当网关发现并将一个子设备连接到网关后，如果需要该子设备能够通过物联网平台进行远程管理，需要对子设备进行认证。认证的目的主要有以下三点：

（1）校验子设备的合法性

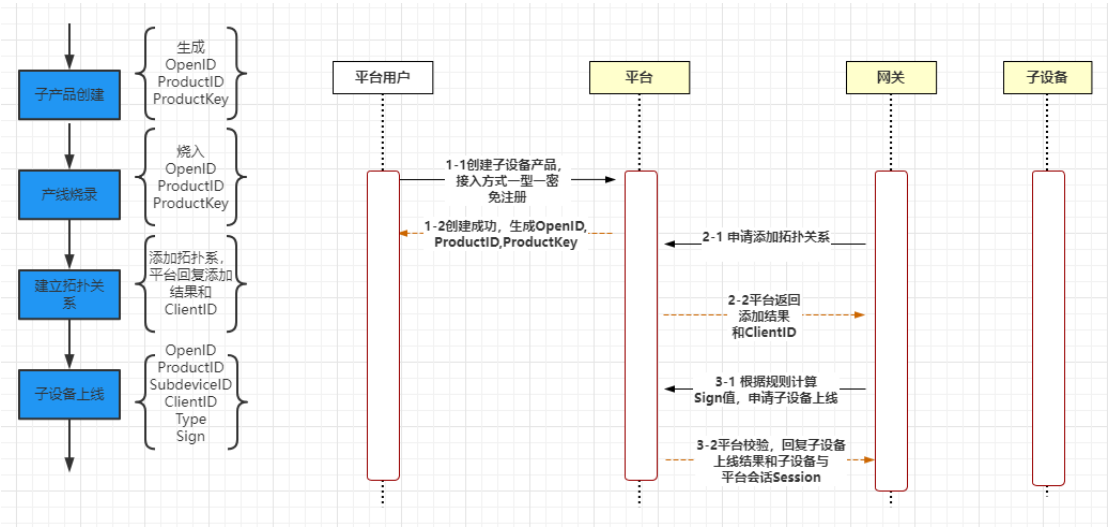
（2）维护网关和子设备的拓扑关系

（3）获取子设备上线所需要的参数

不同类型的子设备认证流程如下：

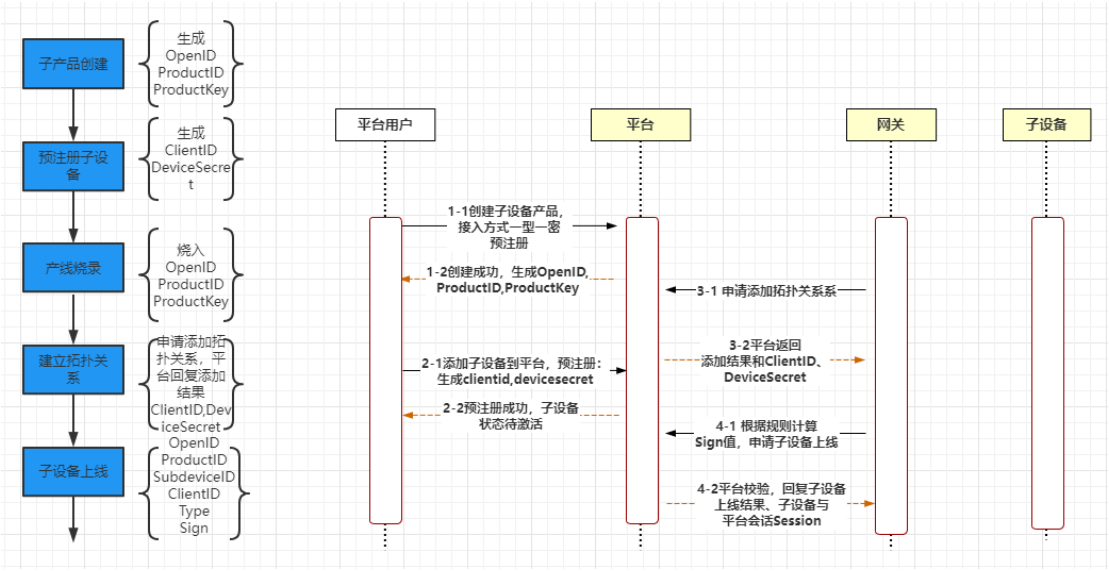
（1）一型一密免注册

一型一密免注册子设备认证流程如图：



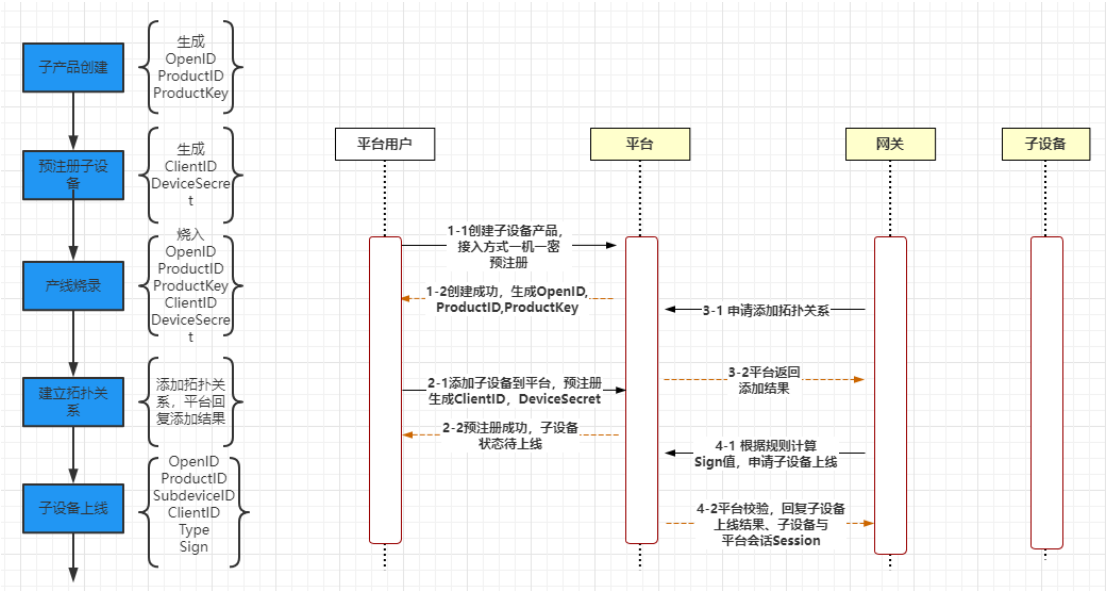

（2）一型一密预注册设备

一型一密预注册子设备认证流程如图：




（3）一机一密预注册设备

一机一密预注册子设备认证流程图




# 1.4 接入协议数据帧格式概述

网关设备接入协议定义的数据格式统一为 json，其中包括但不限于以下属性: HeaderCtrl、Mode、Nonce、HeaderOpts 以及 Payload（内容可自定义）。

协议数据的详细内容如下所示：

```
{
    "MessageId" : 1231,
    "HeaderCtrl":1, //命令码
    "Version":"2.0",//协议版本号
    "Payload":{
        "Subdevices":[
                {
                    "Session":"226717734678623",//会话标识
                    "Payload":{} 子设备数据
                }...
            ]
        }
}
```

协议内容中的各个字段说明如表 2.1 所示。

表 2.1 字段说明

| 字段 | 类型 | 说明 |
|---|---|---|
| HeaderCtrl | int | 协议的控制命令：指令定义详见表 2.2。 |
| Version | String | 协议版本号 |
| ~~Mode~~ | ~~int~~ | ~~应用数据（Payload）的安全模式：Mode 为 0 表示 Payload 数据不加密，建议仅在调试时使用此模式；Mode 为 1 表示 Payload 数据使用 AES128~~ |

| | | |
|---|---|---|
| | | ~~加密传输，产品正式上线时建议使用此模式~~ |
| ~~Nonce~~ | ~~int~~ | ~~随机数：由数据发送方随机生成，智能设备每次发送数据帧都需要生成一个随机数~~ |
| Payload | K | 协议消息头可选域：Payload 的具体内容根据 HeaderCtrl 中的 Command 交互指令而定，详见交互流程介绍 |
| Subdevices | [] | 子设备数据集合 |
| Session | String | 子设备会话标识 |

表 2.2 数据帧 Command 指令说明

| Command | 指令名称 | 说明 |
|---|---|---|
| **智能设备主动发起的 Command** | | |
| 26 | 子设备物模型数据历史上报 | 网关主动向平台上报子设备物模型定义的属性事件等历史数据，数据格式为 JSON |
| 27 | 子设备物模型属性上报（JSON） | 网关主动向平台上报子设备物模型定义的属性，数据格式为 JSON |
| 28 | 子设备物模型事件上报(JSON) | 网关主动向平台上报子设备物模型定义的事件，数据格式为 JSON |
| 29 | 子设备物模型属性上报(HEX) | 网关主动向平台上报子设备物模型定义的属性，数据格式为十六进制 |
| 30 | 子设备物模型事件上报(HEX) | 网关主动向平台上报子设备物模型定义的事件，数据格式为十六进制 |
| **IoT 平台主动发起的 Command** | | |

| | | |
|---|---|---|
| 31 | 子设备物模型服务调用(JSON) | IoT 平台通过指令 31，调用子设备在物模型定义中所定义的服务，数据格式 JSON |
| 32 | 子设备物模型服务调用(HEX) | IoT 平台通过指令 32，调用子设备在物模型定义中所定义的服务，数据格式 HEX |
| 33 | 子设备物模型属性查询( JSON) | IoT 平台通过指令 33，查询子设备的属性值，数据格式 JSON |
| 34 | 子设备物模型属性查询(HEX) | IoT 平台通过指令 34，查询子设备的属性值，数据格式 HEX |
| 35 | 子设备物模型属性设置(JSON) | IoT 平台通过指令 35，设置子设备的属性值，数据格式 JSON |
| 36 | 子设备物模型属性设置(HEX) | IoT 平台通过指令 36，设置子设备在物模型定义中所定义的属性，数据格式 HEX |

## 1.5 子设备与 IOT 平台交互 topic 列表

| | 功能名称 | topic | 功能码 | 方向 |
|---|---|---|---|---|
| 1 | 申请添加拓扑关系 | /sys/${openId}/${productId}/${ gateway_deviceid}/add_subdevice | 无 | 网关—>平台 |
| | | /sys/${openId}/${productId}/${ gateway_deviceid }/add_subdevice_ack | 无 | 平台—>网关 |
| 2 | 子设备上线 | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice_lo | 无 | 网关—>平台 |

| | | | | |
|---|---|---|---|---|
| | | gin | | |
| | | /sys/${openId}/${productId}/ ${ gateway_deviceid }/subdevice_login_ack | 无 | 平台—> 网关 |
| 3 | 子设备下线 | /sys/${openId}/${productId}/ ${ gateway_deviceid }/subdevice_signout | 无 | 网关—>平台 |
| | | /sys/${openId}/${productId}/ ${ gateway_deviceid }/subdevice_signout_ack | 无 | 平台—> 网关 |
| 4 | 网关删除子设备 | /sys/${openId}/${productId}/ ${ gateway_deviceid }/delete_subdevice | 无 | 网关—>平台 |
| | | /sys/${openId}/${productId}/ ${ gateway_deviceid }/delete_subdevice_ack | 无 | 平台—> 网关 |
| 5 | 网关主动查询拓扑关系 | /sys/${openId}/${productId}/ ${ gateway_deviceid }/query_subdevicelist | 无 | 网关—>平台 |
| | | /sys/${openId}/${productId}/ ${ gateway_deviceid }/query_subdevicelist_ack | 无 | 平台—> 网关 |

| | | | | |
|---|---|---|---|---|
| 6 | 平台禁用、启用子设备 | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/disablelist | 无 | 平台—>网关 |
| | | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/disablelist_ack | 无 | 网关—>平台 |
| 7 | 平台添加子设备 | /sys/${openId}/${productId}/${gateway_deviceid}/subdevice/add_subdevice_plat | 无 | 平台—>网关 |
| | | /sys/${openId}/${productId}/${gateway_deviceid}/subdevice/add_subdevice_plat_ack | 无 | 网关—>平台 |
| 8 | 平台删除子设备 | /sys/${openId}/${productId}/${gateway_deviceid}/subdevice/delete_subdevice_plat | 无 | 平台—>网关 |
| | | /sys/${openId}/${productId}/${gateway_deviceid}/subdevice/delete_subdevice_plat_ack | 无 | 网关—>平台 |
| 9 | 平台主动查询拓扑关系 | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevicelist | 无 | 平台—>网关 |
| | | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevicelist_ack | 无 | 网关—>平台 |

| | | | | |
|---|---|---|---|---|
| 10 | 属性上报（json） | /sys/${openId}/${productId}/<br>${ gateway_deviceid }/subdevice/<br>model/property/report | 27 | 网关—>平台 |
| | | /sys/${openId}/${productId}/<br>${ gateway_deviceid }/subdevice/<br>model/property/report_ack | 27 | 平台—><br>网关 |
| 11 | 事件上报（json） | /sys/${openId}/${productId}<br>/${ gateway_deviceid }/subdevice<br>/model/event/report | 28 | 网关—>平台 |
| | | /sys/${openId}/${productId}/<br>${ gateway_deviceid }/subdevice/<br>model/event/report_ack | 28 | 平台—><br>网关 |
| 12 | 历史物模型数据上报 | /sys/${openId}/${productId}/<br>${gateway_deviceid}/subdevice/<br>model<br>/history/property/event/report | 无 | 网关—>平台 |
| | | /sys/${openId}/${productId}/<br>${gateway_deviceid}/subdevice/<br>model<br>/history/property/event/report_ack | 无 | 平台—><br>网关 |
| 13 | 服务调用（json） | /sys/${openId}/${productId}<br>/${ gateway_deviceid }/subdevice<br>/model/service | 31 | 平台—><br>网关 |

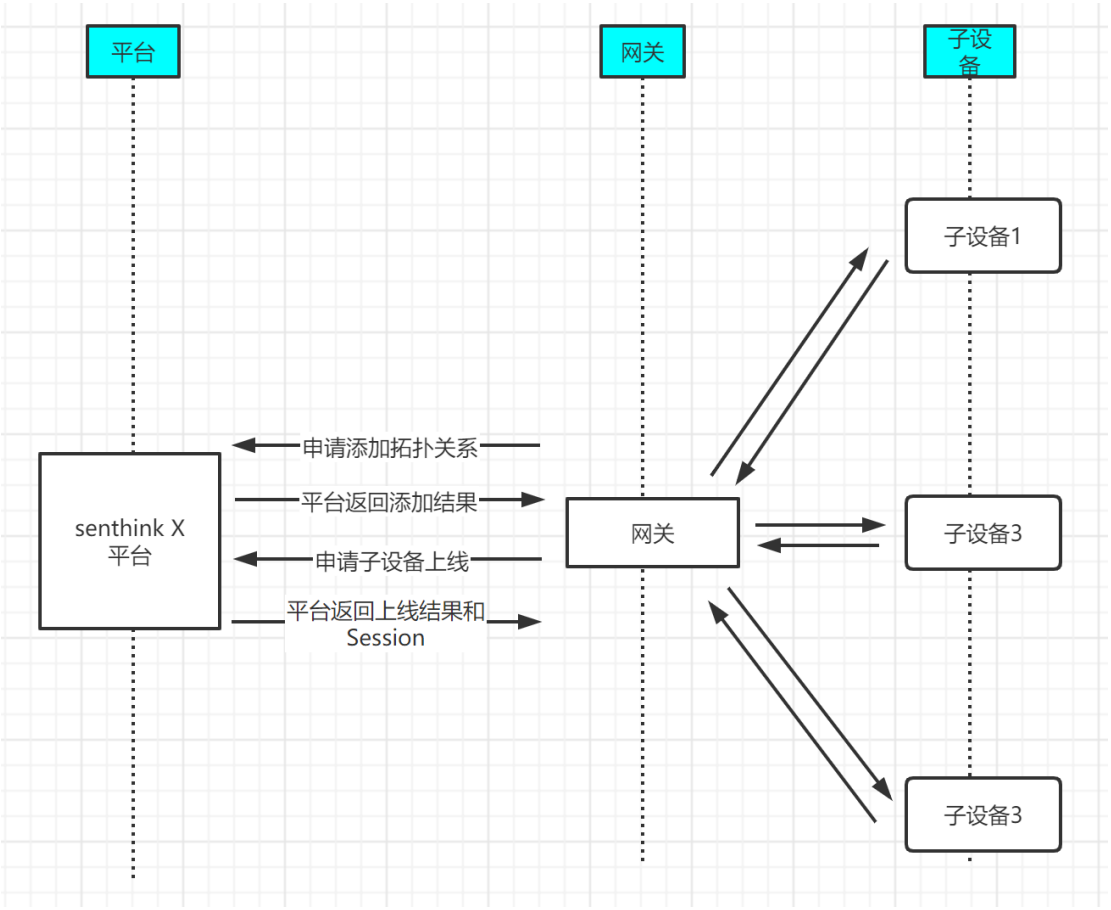
| | | | | |
|---|---|---|---|---|
| | | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/service/ack | 31 | 网关—>平台 |
| 14 | 属性设置（json） | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/set | 35 | 平台—>网关 |
| | | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/set_ack | | 网关—>平台 |
| 15 | 属性查询（json） | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/ask | 33 | 平台—>网关 |
| | | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/report | 27 | 网关—>平台 |
| 16 | 物模型 16 进制上报 | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/hex/report | 29、30 | 网关—>平台 |
| | | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/hex/report_ack | 29、30 | 平台—>网关 |
| 17 | 物模型 16 进制下发 | /sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/ | 32、34、36 | 平台—>网关 |

| | | | | |
|---|---|---|---|---|
| | | model/hex/downlink | | |
| | | /sys/${openId}/${productId}/<br>${ gateway_deviceid }/subdevice/<br>model/hex/downlink_ack | 32、29 | 网关—>平台 |

## 1.6 子设备与 IOT 平台交互流程

在此假设智能设备开发人员已经按照文档的 1.2 子设备接入前准备事项的要求准备好，并确保网关已经认证入网。下面主要描述了，申请添加拓扑关系，子设备上线， 物模型子设备数据交互。 更多详细信息，请查看对应章节。

- 添加网关与子设备拓扑关系




- 物模型设备数据交互

物模型交互分两中，分别为 JSON 格式和 HEX 格式，它们由创建产品的时候定义。 以下为 JSON 格式交互示例。

## 1.6.1 申请添加拓扑关系

申请添加拓扑关系通用 topic：

topic:

/sys/${openId}/${productId}/${gateway_deviceid} /add_subdevice

子设备上报数据之前需要和网关建立拓扑关系。数据格式和内容如下所示：

```
{
```

```
"MessageId" : 1231,
"Version":"2.0",//协议版本号
"Payload":{
        "Subdevices":
            [
                {
                    "OpenID":"E0C040B1",
                     "ProductID":"A4BC6002",
                     "SubdeviceID":"A4BC6002",
                     "Type":0,  //0:一型一密免注册，1：一型一密预注册，2：一机一密预注册
                }
            ]
    }
}
```

网关向 IoT 平台发起请求的数据帧中，每个子设备数据中都有自己的 Payload 域，用于携带额外的信息。Payload 域中包含的内容及说明如下所示：

| Payload | 说明 |
|---|---|
| OpenID | 硬件设备所属厂商的唯一标识 ID，固定 8 位 16 进制字符串并由 IoT 平台分配 |
| ProductID | 硬件设备所属产品的唯一标识 ID，固定 8 位 16 进制字符串并由 IoT 平台分配 |
| SubdeviceID | 硬件设备所属设备的唯一标识 ID，由硬件自行分配，长度不限制 |
| Type | 硬件设备类型，0:一型一密免注册，1：一型一密预注册，2：一机一密预注册 |

平台回复 topic：

/sys/${openId}/${productId}/${gateway_deviceid}/add_subdevice_ack

一型一密免注册 & 一机一密预注册&一型一密预注册:需要返回设备的 ClientID 和 DeviceSecret。

响应设备的数据帧格式以及内容如下表所示：

```
{
    "MessageId" : 1231,
    "Version":"2.0",//协议版本号
    "Payload":{
        "Subdevices":
            [
                {
                    "OpenID ":"E0C040B1",
                     "ProductID":"A4BC6002",
                     "SubdeviceID":"A4BC6002",
                    "Payload":{
                        "Code":0,
                        "ClientID":"1234567890....",   //32 位 UUID
                        "DeviceSecret":"FE448731"
                    }
                }
            ]
    }
}
```

Code 属性说明：

**对于子设备申请拓扑关系，Code 定义为平台回复设备的认证请求的响应码。响应码定义如下表所示：**

| 响应码（Code） | 响应码的说明 |
| --- | --- |
| 0 | 请求处理成功 |
| 1 | 请求处理失败：IoT 平台系统繁忙，暂时无法添加请求。**建议智能设备重试。** |
| 2 | 请求处理失败：申请所使用的产品未注册，无法处理认证请求，或者免注册设备首次上报设备号有误。**建议检查 OpenID、ProductID、SubdeviceID，是否正确。** |
| 4 | 认证请求处理失败：**该设备已被禁用，预注册设备不存在，** |
| 5 | 参数 type 类型错误或者 type 类型未定义，请检查 type 类型和平台产品 type 定义一致 |

为了保证设备端在整个平台的 ClientID 的唯一性，在认证成功之后，使用认证结果中的 ClientID 进行后面的上线操作。

## 1.6.2 子设备上线

子设备上线 topic：

topic：

/sys/${openId}/${productId}/${gateway_deviceId } /subdevice_login

子设备和网关在成功建立拓扑关系，且子设备与网关的状态变为已关联后，必须向平台发送请求上线数据帧，以让平台为子设备生成会话 Session；否则平台将不处理该子设备的数据。

数据帧格式和内容如下所示：

```
{
    "MessageId" : 1231,
    "Version":"2.0",//协议版本号
    "Payload":{
```

```
            "Subdevices":
                [
                    {
                        "OpenID":"E0C040B1",
                        "ProductID":"A4BC6002",
                        "SubdeviceID":"....", //设备标识
                        "ClientID":"....", //认证后返回的 ClientID
                        "Type":0,   //0:一型一密免注册，
                        "Sign":"FE448731",
                        "Timestamp":"1669107618",   //当前时间戳,秒
                    }
                ]
        }
    }
```

网关向 IoT 平台发起的子设备上线请求数据帧中包含每个子设备的 Payload 域，用于携带子设备的额外的信息。Payload 域中包含的内容及说明如下所示：

| Payload | 说明 |
|---|---|
| OpenID | 硬件设备所属厂商的唯一标识 ID，固定 8 位 16 进制字符串并由 IoT 平台分配 |
| ProductID | 硬件设备所属产品的唯一标识 ID，固定 8 位 16 进制字符串并由 IoT 平台分配 |
| SubdeviceID | 智能硬件设备的唯一标识 ID，可以根据不同的应用场景，由 IoT 平台、硬件设备所属厂商或者应用集成商来分配； |
| Type | 硬件设备类型，0:一型一密免注册，1：一型一密预注册，2：一机一密预注册 |

| | |
|---|---|
| Sign | 子设备向 IoT 平台请求上线的时候，使用加密算法对特定字段进行加密运算所生成的数据，提交给平台。平台据此判断是否允许该设备上线 |

平台收到子设备上线请求后，根据每个子设备的当前产品类型是免注册还是预注册，重新计算 Sign 并与设备上报的 Sign 值进行对比，不同类型设备的 Sign 计算规则如下：

1、一型一密免注册 or 一型一密预注册 or 一机一密预注册

智能设备使用子设备的 ProductKey 对数据帧中的 OpenID、ProductID、SubdeviceID 以及 timestamp 的拼接结果进行加密运算，截取运算结果的高 4 bytes 作为 Sign 的值。

加密运算方法如下（伪代码）：

加密方式：AES128_ECB_PKCS5Padding

Sign = ENCRYPT( DeviceSecret , ProductID.SubdeviceID__ProductKey_timeStamp );

示例如下：

DeviceSecret: A5E8DA27BA1A2780669A6BAACCA95679

ProductKey ：8CC530001D407E99CD4D48F2D8F6D9E9

ProductID：0F18D446

SubdeviceID：20MQTTJ000263

Timestamp:1696767676

计算过程为：

ENCRYPT("A5E8DA27BA1A2780669A6BAACCA95679","0F18D446.20MQTTJ000263_8CC530001D407E99CD4D48F2D8F6D9E9_1696767676".getBytes(StandardCharsets.UTF_8)).toHexString().toUpperCase();

则计算结果为：

21FB7EE7A2F32C32AA2C5143E2A2DCA04E60EDE6146066E80EE64492D8376D956BC0885DDCD1CF31AE1002D43BBC92BB5989DF9E706EB27C03CF8709D4011670FFBD06BEEBBBC1E5570252F29DF5D0FF

取最低 4bytes 为：9DF5D0FF

IoT 平台收到子设备的上线请求数据帧并成功认证子设备身份后，会为该子设备生成会话标识。

平台回复 topic：

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice_login_ack

响应智能设备入网请求的数据如下：

```
{
    "MessageId" : 1231,
    "Version":"2.0",//协议版本号
    "Payload":{
        "Subdevices":
            [
                {
                    "ClientID":"....", //认证后返回的 ClientID
                    "Payload":{
                        "Code":0,
                        "Session":" 226717734678623"
                    },
                    "Timestamp":"1669107618" //激活时上送的 timestamp
                }
            ]
    }
}
```

Code 说明：

对于子设备上线指令，Code 定义为平台回复子设备上线请求的响应码。响应码定义如下表所示：

| 响应码<br>(Code) | 响应码的说明 |
|---|---|
| 0 | 入网请求处理成功：表示 IoT 平台成功处理请求并成功为子设备成功生成会话标识。 |
| 1 | 入网请求处理失败：IoT 平台系统繁忙，暂时无法处理上线请求，或者子设备与网关尚未添加拓扑关系。**建议智能设备重试。** |
| 2 | 入网请求处理失败：入网所使用的产品未注册，无法处理上线请求。**建议检查 OpenID 和 ProductID 是否正确。** |
| 3 | 入网请求处理失败：设备入网请求的签名（Sign）值不正确。**建议检查 ProductKey 是否正确，或者签名算法是否使用正确。** |
| 4 | 认证请求处理失败：**该设备已被禁用，设备不存在，** |
| 5 | 参数 Type 错误，请检查 Type 参数和，后台产品 Type 一致 |

**只有响应码（Code）值为 0 的时候**，响应数据中的 Payload 中所携带的 Session 才是 IoT 平台分配的**有效的 Session**：

| 字段 | 说明 |
|---|---|
| Session | 后续子设备跟 IoT 平台进行通信的会话。 |

## 1.6.3 子设备下线

子设备下线 topic：

topic：

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice_signout

网关监听到子设备下线后，须向平台发送子设备离线数据帧。

数据帧格式和内容如下所示：

```
{
    "MessageId" : 1231,
    "Version":"2.0",//协议版本号
    "Payload":{
            "Subdevices":
                [
                    {
                        "Session":" 226717734678623",
                        "Payload":{ }
                    }
                ]
    }
}
```

服务端收到子设备下线请求之后会通过如下 topic 进行应答 ack。平台将子设备状态改为离线。

平台回复 topic:

topic:
/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/subdevice_signout_ack

数据内容格式如下：

```
{
    "MessageId":1231,
```

```
"Version":"2.0",//协议版本号
"Payload":{
        "Subdevices":
            [
                {
                    " Session ":"226717734678623",
                    "Code":"200",
                    "Msg":"SUCCESS",
                    "Data":{}      // 非必须字段
                } …
            ]
        }
}
```

### 1.6.4 网关添加子设备

网关添加子设备同流程。需先平台申请添加扩扑关系，平台验证并添加完成后，将结果应答给网关，网关再发起子设备上线操作

### 1.6.5 网关删除子设备拓扑关系

网关删除子设备 topic:

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/delete_subdevice

网关删除子设备后，必须向平台发送删除子设备数据帧请求。数据帧以 JSON 格式上报给平台。数据内容格式如下:

```
{
```

```
    "MessageId": 192155183,
    "Version":"2.0",//协议版本号
    "Payload": {
                "Subdevices":
                    [{
                        "OpenID":"E0C040B1",
                        "ProductID":"A4BC6002",
                        "ClientID":"....", //认证后返回的 ClientID
                     }]
    }

}
```

平台回复 topic：

/sys/${openId}/${productId}/${ gateway_deviceId }/delete_subdevice_ack

数据内容格式如下：

```
{
        "MessageId":1231,
        "Version":"2.0",//协议版本号
        "Payload": {
            "Code":"200",
            "Msg":"SUCCESS",
            "Data":{}   // 非必须
        }
}
```

### 1.6.6 网关主动查询拓扑关系

网关主动查询拓扑关系 topic:

/sys/${openId}/${productId}/${ gateway_deviceid } /query_subdevicelist

网关主动查询拓扑关系，查询信息以 JSON 格式发给平台。平台收到查询指令，应答拓扑关系。数据内容格式如下：

```
{
    "MessageId": 192155183,
    "Version":"2.0",//协议版本号
    "Payload": {}
}
```

平台回复 topic

/sys/${openId}/${productId}/${gateway_deviceId } /query_subdevicelist_ack

数据内容格式如下：

```
{
        "MessageId": 192155183,
        "Version":"2.0",//协议版本号
            "Payload":{
                "Subdevices":
                    [
                        {
                            "OpenID":"E0C040B1",
```

```
                                "ProductID":"A4BC6002",
                                "SubdeviceID":"....", //设备的 SubdeviceID
                                "ClientID":"....", //认证后返回的 ClientID
                                "Online": 0   // 0 离线   1 在线 -1 未激活
                                "State":0   // 0 禁用   1 启用
                                "RelationState":2,//子设备与网关的关联关系：0 未关联、1 已关联、2 解除中

                            }  ...
                        ]
                }
    }
```

## 1.6.7 平台禁用、启用子设备

平台禁用、启用子设备 topic:

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/disablelist

平台将子设备禁用、启用的信息以 JSON 格式下发给网关。网关根据下发指令，执行操作。数据内容格式如下：

```
{
    "MessageId": 192155183,
    "Version":"2.0",//协议版本号
    "Payload": {
                "Subdevices":
```

```
                    [
                        {
                            "OpenID":"E0C040B1",
                        "ProductID":"A4BC6002",
                        "SubdeviceID":"A4BC6002" ,
                        "State":0   // 0 禁用   1 启用
                        },
                        {
                            "OpenID":"E0C040B1",
                        "ProductID":"A4BC6002",
                        "SubdeviceID":"A4BC6002" ,
                        "State":1   // 0 禁用   1 启用
                        }
                          ]
    }
}
```

网关回复 topic

topic :

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/disablelist_ack

数据内容格式如下:

```
 {
        "MessageId": 192155183,
        "Version":"2.0",//协议版本号
            "Payload":{
```

```
                "Subdevices":
                    [
                        {
                            "OpenID":"E0C040B1",
                            "ProductID":"A4BC6002",
                            "SubdeviceID":"A4BC6002" ,
                                                    "State":0   // 0 禁用   1 启用
                        },
                        {
                            "OpenID":"E0C040B1",
                            "ProductID":"A4BC6002",
                            "SubdeviceID":"A4BC6002" ,
                                                    "State":1   // 0 禁用   1 启用
                        }  ...
                    ]
            }
 }
```

### 1.6.8 平台通知网关添加子设备（白名单）

平台添加子设备 topic:

topic:

/sys/${openId}/${productId}/${gateway_deviceid}/subdevice/add_subdevice_plat

平台添加子设备的信息以 JSON 格式下发给网关。网关根据下发指令，执行操作，平台收到回复后对结果已知晓，不做处理。数据内容格式如下:

```
{
```

```
    "MessageId": 192155183,
    "Version":"2.0",//协议版本号
    "Payload": {
                "Subdevices":
                    [
                        {
                            "OpenID":"E0C040B1",
                            "ProductID":"A4BC6002",
                            "SubdeviceID":"....",
                        },
                        {
                            "OpenID":"E0C040B2",
                            "ProductID":"A4BC6003",
                            "SubdeviceID":"....",
                        }
                    ]
    }
}
```

网关回复 topic

Topic:

/sys/${openId}/${productId}/${gateway_deviceid}/subdevice/add_subdevice_plat_ack

数据内容格式如下：

```
 {
```

```
"MessageId": 192155183,
"Version":"2.0",//协议版本号
    "Payload":{
        "Subdevices":
            [
                {
                    "OpenID":"E0C040B1",
                    "ProductID":"A4BC6002",
                    "SubdeviceID":"....",
                    "Result": 0 // 0  添加失败   1  添加成功
                },
                {
                    "OpenID":"E0C040B2",
                    "ProductID":"A4BC6003",
                    "SubdeviceID":"....",
                    "Result": 0 // 0  添加失败   1  添加成功
                }  ...
            ]
    }
}
```

### 1.6.9 平台删除子设备

平台删除子设备 topic:

topic:

/sys/${openId}/${productId}/${gateway_deviceid}/subdevice/delete_subdevice_plat

平台删除子设备的信息以 JSON 格式下发给网关。网关根据下发指令，执行操作。数据内容

格式如下：

```
{
    "MessageId": 192155183,
    "Version":"2.0",//协议版本号
    "Payload": {
                "Subdevices":
                    [
                        {
                            "OpenID":"E0C040B1",
                            "ProductID":"A4BC6002",
                            "SubdeviceID":"A4BC6003"
                        },
                        {
                            "OpenID":"E0C040B2",
                            "ProductID":"A4BC6003",
                            "SubdeviceID":"A4BC6003"
                                                }
                                            ]
            }
}
```

网关回复 topic

Topic:

/sys/${openId}/${productId}/${gateway_deviceid}/subdevice/delete_subdevice_plat_ack

数据内容格式如下：

```
{
        "MessageId": 192155183,
        "Version":"2.0",//协议版本号
            "Payload":{
                "Subdevices":
                    [
                        {
                            "OpenID":"E0C040B1",
                            "ProductID":"A4BC6002",
                            "SubdeviceID":"A4BC6002" ,
                            "Result": 0 // 0 删除失败  1 删除成功
                        },
                        {
                            "OpenID":"E0C040B1",
                            "ProductID":"A4BC6002",
                            "SubdeviceID":"A4BC6003" ,
                            "Result": 1 // 0 删除失败  1 删除成功
                        }
                    ]
            }
}
```

### 1.6.10 平台主动查询拓扑关系（预留）

平台查询拓扑关系 topic：

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/subdevicelist

平台主动查询网关与子设备的拓扑关系，查询指令以 JSON 格式下发给网关。网关根据下发指令，应答当前拓扑关系。数据内容格式如下：

```
{
    "MessageId" : 1231,
    "Version":"2.0",//协议版本号
    "Payload": {}
}
```

网关回复 topic:

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/subdevicelist_ack

数据内容格式如下：

```
{
    "MessageId": 1231,
    "Version":"2.0",//协议版本号
        "Payload":{
            "Subdevices":
                [
                    {
                                "OpenID":"E0C040B1",
                                "ProductID":"A4BC6002",
                        "SubdeviceID":"A4BC6002" .
                        "Online": 0   // 0 离线   1 在线
```

```
                    "State":0   // 0 禁用   1 启用
                },{
                        "OpenID":"E0C040B1",
                         "ProductID":"A4BC6002",
                    "SubdeviceID":"A4BC6002"
                    ""Online": 0   // 0 离线   1 在线
                    "State":0   // 0 禁用   1 启用
                }  …
            ]
        }
}
```

## 1.6.11 物模型产品数据交互

### 1.6.11.1 JSON 格式数据交互

前提: 后台创建网关产品、子设备产品的时候选择为物模型，数据格式选择为 JSON 格式。产品定义物模型，后面子设备属性上报，事件上报，服务调用都是基于定义的物模型。

#### 1.6.11.1.1 属性上报

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/report

网关将所有子设备数据转成指定 json 格式上报给平台。上行数据帧使用的 Command 为 27。数据内容格式如下

```
{
"MessageId": 1231,
```

```
"Version":"2.0",//协议版本号
"HeaderCtrl":27, //命令码
"Payload":{
        "Subdevices":
            [
                {
                        "ClientID":"A4BC6002",//子设备 ClientID
                        "Session":"226717734678623",//会话标识
                        "Payload":{
                            "temp":100,   // 属性
                            "humi":20     // 属性
                        }
                },{
                        "ClientID":"A4BC6003",//子设备 ClientID
                        "Session":"226717734678624",//会话标识
                        "Payload":{
                            "temp":100,   // 属性
                            "humi":20     // 属性
                        }
                }...
            ]
    }
}
```

服务端收到属性上报之后会通过如下 topic 进行应答 ack。

topic:
/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/report_ack

数据内容格式如下:

```
{
    "MessageId":1231,
    "Version":"2.0",//协议版本号
    "HeaderCtrl":27,
    "Payload": {
        "Code":"200",
        "Msg":"SUCCESS",
        "Data":{}   // 非必须
    }
}
    //备注：目前统一返回成功，具体的错误记在日志里
```

### 1.6.11.1.2 事件上报

topic:
/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/event/report

网关将所有子设备数据转成指定 json 格式上报给平台。上行数据帧使用的 Command 为 28。数据内容格式如下

```
{
```

```
"MessageId": 1231
"HeaderCtrl":28,
"Version":"2.0",//协议版本号
"Payload":{
        "Subdevices":
            [
                {
                        "ClientID":"A4BC6002",//子设备 ClientID
                        "Session":"226717734678623",//会话标识
                        "Identifier":"event.identifier" .
                        "Payload":{
                            "temp":100,   // 属性
                            "humi":20     // 属性
                        }
                    } , {
                        "ClientID":"A4BC6003",//子设备 ClientID
                        "Session":"226717734678624",//会话标识
                        "Identifier":"event.identifier",
                        "Payload":{
                            "temp":100,   // 属性
                            "humi":20     // 属性
                        }
                    }...
                ]
}
```

```
}
```

**注意**: event.identifier 为子设备对应产品物模型的事件标识符。

服务端收到事件上报之后会通过如下 topic 进行应答 ack。

topic:
/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/event/report_ack

数据内容格式如下:

```
{
    "MessageId":1231,
    "Version":"2.0",//协议版本号
    "HeaderCtrl":28,
    "Payload": {
        "Code":"200",
        "Msg":"SUCCESS",
        "Data":{}   // 非必须
    }
}
    //备注：目前统一返回成功，具体的错误记在日志里
```

### 1.6.11.1.3 历史物模型数据上报

topic:
sys/${openId}/${productId}/${gateway_deviceid}/subdevice/model/history/data/report

网关将子设备历史物模型数据转成指定 json 格式上报给平台。数据内容格式如下

```
{
    "MessageId":"123",
    "HeaderCtrl":26,
    "Version":"2.0",
    "Payload":{
        "Subdevices":[
            {
                "ClientID":"318FE7A2F7D593A72zishebei",
                "Session":"7E7AE997",
                "Payload":[
                    {
                        "properties":[
                            {
                                "Power":{
                                    "value":"3",
                                    "time":1670477584000
                                },
                                "WF":{
                                    "value":"3",
                                    "time":1670477584000
                                }
                            },
                            {
```

```
                    "Power":{
                        "value":"3",
                        "time":1670477584000
                    },
                    "WF":{
                        "value":"3",
                        "time":1670477584000
                    }
                }
            ],
            "events":[
                {
                    "alarmEvent":{
                        "value":{
                            "Power":"3",
                            "WF":"3"
                        },
                        "time":1670477584000
                    }
                }
            ]
        },
        {
            "properties":[
```

```
                {
                    "Power":{
                        "value":"3",
                        "time":1670473984000
                    },
                    "WF":{
                        "value":"3",
                        "time":1670473984000
                    }
                }
            ],
            "events":[
                {
                    "alarmEvent":{
                        "value":{
                            "Power":"3",
                            "WF":"3"
                        },
                        "time":1670473984000
                    }
                }
            ]
        }
    ]
```

```
            }
        ]
    }
    }
```

服务端收到历史属性上报之后会通过如下 topic 进行应答 ack。

topic:

sys/${openId}/${productId}/${gateway_deviceid}/subdevice/model/history/data/report_ack

```
{
    "MessageId":1231,
    "Version":"2.0",
    "Payload":{
    "Code":"200",
    "Msg":"SUCCESS",
    "Data":{}   // 如果有输出参数，则对应 Data 字段， 若无，可为空。
    }
}
```

### 1.6.11.1.4 服务调用

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/service

平台根据后台子设备的物模型定义的服务，将服务指令以 json 格式下发给网关。网关根据下发指令，匹配对应子设备，执行对应的服务操作。下行数据帧使用的 Command 为 31。

```
{
```

```
"MessageId": 1231
"Version":"2.0",//协议版本号
"HeaderCtrl": 31, //命令码
"Payload":{
        "Subdevices":
            [
                {
                    "ClientID":"A4BC6002",//子设备 ClientID
                    "Session":"226717734678623",//会话标识
                  "Payload":{
                        "serviceIdentify":"switch",   // 服务标识
                        "params":{
                              "on":1   // 输入参数
                              }
                      }
                }
                }
            ]
    }
}
```

网关通过如下 topic 进行应答 ack。

topic:
/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/service/ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "Version":"2.0",//协议版本号
   "HeaderCtrl": 31, //命令码
    "Payload":{
            "Subdevices":
                [
                    {
                        "ClientID":"A4BC6002" ,//子设备 ClientID
                        "Session":"226717734678623" ,//会话标识
                      "Code":"200",
                        "Msg":"SUCCESS",
                        "ServiceIdentify":"switch",
                        "Data":{} // 非必须， 如果有输出参数，则对应 Data 字段， 若无，可为空
                    }
                ]
    } // 非必须， 如果有输出参数，则对应 Data 字段， 若无，可为空。
}
```

其中 Data 字段为非必填字段，对应物模型定义的时候的输出参数，如果非空，则 Data 的数据为 JSON 格式。

### 1.6.11.1.5 属性设置

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/propert

y/set

平台根据网关上报上来的子设备的属性，可将对应的属性设置指令以 json 格式下发给网关。网关根据下发指令，匹配对应子设备，执行属性设置操作。下行数据帧使用的 Command 为 35，数据内容格式如下：

```
{
"MessageId": 192155183
"HeaderCtrl": 35,
"Version":"2.0",//协议版本号
"Payload": {
                "Subdevices":
                    [{
                        "ClientID":"A4BC6002",//子设备 ClientID
                        "Session":"226717734678623",//会话标识
                         "Payload": {
                              "params": [{
                                        "identify": "temp",    // 属性标识
                                        "value": "100"          // 属性值
                                   }]
                         },
                     }]
}
```

网关收到属性设置指令之后会通过属性上报 topic 进行应答 ack。

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/set_ack

数据内容格式如下：

```
{
    "MessageId": 1231,
    "Version":"2.0",//协议版本
    "HeaderCtrl": 35,
    "Payload": {
        "Subdevices": [
            {
                "ClientID": "A4BC6002",
                "Session": "226717734678623",
                "Data": [
                    {
                        "PropertyIdentify": "prop_int",
                        "Code": "200",
                        "Msg": "SUCCESS"
                    },
                    {
                        "PropertyIdentify": "prop_bool",
                        "Code": "400",
                        "Msg": "Fail"
                    }
                ]
```

```
            }
        ]
    }
}
```

### 1.6.11.1.6 属性查询

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/ask

平台根据网关上报的子设备的属性，可将对应的属性查询指令以 json 格式下发给网关。网关根据下发指令，匹配对应子设备，执行对应的属性查询操作。下行数据帧使用的 Command 为 33，数据内容格式如下：

```
{
    "MessageId": -192155183
    "Version":"2.0",//协议版本号
    "HeaderCtrl": 33,
    "Payload": {
        "Subdevices":
            [
                {
                    "ClientID":"A4BC6002",//子设备 ClientID
                    "Session ":"226717734678623",//会话标识
                     "Payload": {
```

```
                                    "params": ["temp"]
                                }
                    }]
        },
}
```

网关收到属性查询指令之后会通过属性上报 topic 进行应答 ack。

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/property/report

数据内容格式如下:

```
{
 "MessageId": 192155183
 "Version":"2.0",//协议版本号
 "HeaderCtrl":27, //命令码
 "Payload":{
            "Subdevices":
                [
                    {
                        "ClientID":"A4BC6002",//子设备 ClientID
                        "Session":"226717734678623",//会话标识
                        "Payload":{
                            "temp":100   //属性
```

```
                    }
                }
            ]
        }
    }
```

### 1.6.11.2 HEX 格式数据交互

网关数据为 JSON 格式，子设备数据中 payload 部分为 HEX

**属性上报**和**事件上报**都使用 16 进制上报 topic 进行上报,通过 HeaderCtrl 区分，结合解析脚本对子设备数据中 payload 部分进行 16 进制编码之后上报。

**属性设置、属性查询**和**服务调用**通过物模型 16 进制 topic 下发,同样设备收到的数据中子设备数据中 payload 部分也是经过解析脚本编码之后的 16 进制格式数据。

#### 1.6.11.2.1 物模型 16 进制上报

topic:
/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/hex/report

事件上报和属性上报使用相同的 topic， 通过 HeaderCtrl 来对属性上报和事件上报进行区分。

| HeaderCtrl | 说明 |
| --- | --- |
| 29 | 属性上报 |
| 30 | 事件上报 |

```
{
"MessageId": 1231,
```

```
"Version":"2.0",//协议版本号
"HeaderCtrl":29, //命令码
"Payload":{
        "Subdevices":
            [
                {
                    "ClientID":"A4BC6002" ,//子设备 ClientID
                    "Session":"226717734678623",//会话标识
                    "Payload":"0101002d0142f6e76d"
                },{
                    "ClientID":"A4BC6003",//子设备 ClientID
                    "Session":"226717734678624",//会话标识
                    "Payload":"0101002d0142f6e76d"
                }...
            ]
    }
}
```

服务端收到属性上报之后会通过如下 topic 进行应答 ack。

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/hex/report_ack

数据内容格式如下:

```
{
    "MessageId":1231,
    "Version":"2.0",//协议版本号
    "HeaderCtrl":29,
    "Payload": {
        "Code":"200",
        "Msg":"SUCCESS",
        "Data":{}  // 非必须
    }
}
//备注：目前统一返回成功，具体的错误记在日志里
```

### 1.6.11.2.2 物模型 16 进制下发

topic:
/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/hex/downlink

属性设置、属性查询和服务调用使用相同的 topic， 通过 HeaderCtrl 来对属性设置和服务调用来进行区分。

| HeaderCtrl | 说明 |
|---|---|
| 32 | 服务调用 |
| 34 | 属性查询 |
| 36 | 属性设置 |

```
{
"MessageId": 192155183
"Version":"2.0",//协议版本号
"HeaderCtrl": 32,
"Payload": {
                "Subdevices":
                    [
                        {
                            "ClientID":"A4BC6002",//子设备 ClientID
                            "Session":"226717734678623",//会话标识
                             "Payload": "0101002d0142f6e76d"
                        },{
                            "ClientID":"A4BC6003" ,//子设备 ClientID
                            "Session":"226717734678624",//会话标识
                            "Payload":"0101002d0142f6e76d"
                        }...
                    ]
}
```

网关收到属性设置指令之后会通过属性上报 topic 进行应答 ack。

topic:

/sys/${openId}/${productId}/${ gateway_deviceid }/subdevice/model/hex/downlink_ack

数据内容格式如下:

```
{
"MessageId": 192155183
"Version":"2.0",//协议版本号
"HeaderCtrl":29, //命令码
"Payload":{
                "Subdevices":
                    [
                        {
                            "ClientID":"A4BC6002" ,//子设备 ClientID
                            "Session":"226717734678623",//会话标识
                             "Payload": "0101002d0142f6e76d"
                        }, {
                            "ClientID":"A4BC6003" ,//子设备 ClientID
                            "Session":"226717734678624",//会话标识
                            "Payload":"0101002d0142f6e76d"
                        }...
                    ]
        }
}
```