杭州贤芯科技有限公司

参考文档

HANGZHOU SENTHINK SCIENCE & TECHNOLOGY CO., LTD

文档编号：

保密级别：内部文档，严禁外传

版本编号：V1.0

利尔达物联网全连接云平台

-MQTT 智能设备接入协议文档(V2.0)

修改历史

编制		编制日期		
序号	修改日志	版本	修改人	修改日期
1	新建文档	V1.0	yxf@lierda.com	2023/3/17

目录

1.1 目的

本文档描述了利尔达物联网全连接云平台（简称：X 平台）智能设备 MQTT 接入协议格式内容以及智能硬件设备与 X 平台的交互流程。

1.2 文档范围

本文档从内容上涵盖了智能硬件设备与 X 平台交互的协议格式以及相关通信约定。

1.3 预期读者

嵌入式软件工程师、服务器端开发人员以及软件和硬件测试人员。

1.4 名词术语

名词	解释	备注
AEP/AEP 平台	X 平台中的应用使能平台模块，主要负责与客户对接，提供应用使能服务	
CMP/CMP 平台	X 平台中的连接管理模块，包含 MQTT 连接管理、TCP 连接管理	
OpenID	平台用户唯一 ID	
ProductID	产品唯一 ID	
ProductKey	产品密钥	
NodeEui/DeviceID	设备唯一编号	
DeviceSecret	设备密钥	
SessionID	加密设备认证后平台回复给设备的 sessionID，智能设备可通过约定的加密算法计算 SessionKey	
SessionKey	用于设备上下行数据加密传输时的密钥	

2 IoT 平台概述

图 2.1 IoT 平台架构图

图 2.1 是 X 平台的架构图。X 平台作为一个下层智能硬件设备和上层应用服务的连接桥梁，对智能硬件设备提供一套安全、完善的接入协议，帮助智能硬件设备快速接入云端；同时对上层应用提供标准化、多样化的接口，以帮助客户将智能设备和自己的应用进行连接。

2.1 智能设备接入前准备事项

智能设备在接入 X 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创建一个产品，云平台会为该产品分配一个产品的唯一标识 ProductID 以及产品的密钥 ProductKey；
3. 智能设备接入 X 平台之前，必须要将基础参数烧录进设备。后面设备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：

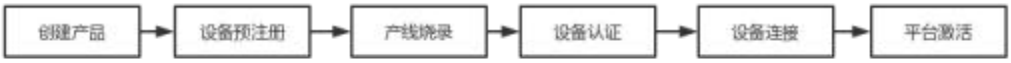
（1）一型一密免注册设备：OpenID、ProductID、ProductKey、NodeEui(设备 ID)

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

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

至此，接入前的准备工作完成！

2.2 智能设备接入流程示例



智能硬件设备使用 MQTT 协议接入 X 平台的流程如上，下面详细进行介绍：

2.2.1 创建产品

在平台进行产品创建后，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连接。

2.2.2 设备预注册

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，免注册设备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，在完成产品创建后设备参数烧录之前，需要将设备预先注册到平台中，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 NodeEui/Deviceid）。完成预注册的设备信息此时已录入平台，设备处于未激活状态，等待连接后自动激活。

2.2.3 产线烧录

智能设备接入 X 平台之前，必须要将基础参数烧录进设备。后面设备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：OpenID、ProductID、ProductKey、NodeEui(设备 ID)

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

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

2.2.4 一型一密设备认证

设备认证为一型一密免注册、一型一密预注册设备才有的流程，一机一密预注册设备直接跳过此流程即可。

MQTT 智能设备在入网之前必须先进行设备认证流程，通过通用账号密码连接 X 平台的 MQTT 服务器进行设备认证。设备认证的目的主要有以下两点：

（1）设备校验平台是否可信，平台检查设备是否合法

（2）设备获取入网所需参数

由上可见，为保证设备安全性并且能够顺利完成入网流程，必须先进行设备认证。智能设备向 MQTT 服务器建立连接并发起认证请求的通用账号密码生成规则如下：

MqttClientID：${ProductID}-${NodeEui(设备 ID)}

MqttUserName：AUTH-${OpenID}-${ProductID}

MqttPassword：${ProductKey}

通过此规则成功登录 MQTT 服务器后的设备权限如下：

- 发布权限 topic：“/sys/device/auth”
- 订阅权限 topic：“/sys/${OpenID}/${ProductID}/${NodeEui}/downlink”

设备完成认证流程后主动断开连接，并再次发起入网连接。此处仅作流程介绍，更详细的设备认证流程请参考 4.1 智能设备认证 IOT 平台流程。

2.2.5 设备连接

MQTT 设备在完成认证流程后，立即断开连接，并根据 MQTT 智能设备入网请求账号密码生成规则生成登录 MQTT 服务器所需的账号信息，重新向 MQTT 发起登录请求进行设备连接请求，设备连接的目的：

（1）平台校验设备合法性后激活设备信息。

（2）使用设备独有的账号密码连接服务器进行数据交互

设备连接信息如下所示：

MqttClientID：V2_ClientId_type_Timestamp
MqttUserName：OpenID-ProductID-NodeEui
MqttPassword：ENCRYPT(DeviceSecret,content)

MqttClientID：格式中| |内为扩展参数。

ClientId：${ProductID}.${NodeEui(设备 ID)}

Type：是否需要校验时间戳的标志位，0-不校验，1-校验。

Timestamp：表示当前时间秒值。时间戳范围：大于当前设备离线时间，小于等于当前时间。

备注：平台端添加预注册设备时默认生成一个 type=0 的 MqttClientID，不校验时间戳，设备端如果需要校验时间戳，那么设置 type=1，password 根据时间戳计算后连接 mqtt broker 认证即可。

MqttPassword：content 的值为提交给服务器的参数（ClientId、ProductKey 和 Timestamp），按照参数名称首字母字典排序， 然后将参数值依次拼接，计算的时候需要将字符串转换成 hex 计算。

PS：可访问参数生成工具，填写注册设备后生成的信息，自动生成设备连接鉴权所需的参数。

Mqtt broker 连接参数计算如下所示：

加密方式：AES128_ECB_PKCS5Padding
输入key：Hex
输入加密内容块：UTF-8
输出内容：Hex并转换为大写

1、type=0

假设 openId=7ED01E2A,productId=AA30C505,nodeEui=20MQTTJ000121， timestamp=1675404867,type=0,deviceSecret=326086233FAB8A96566796551EB1DFF8

那么 mqtt 连接参数如下：

MqttClientID:V2_AA30C505.20MQTTJ000121_0_1675404867

MqttUserName：7ED01E2A-AA30C505-20MQTTJ000121

MqttPassword:5EB6BBC8DE5233E632389FA3E8E133F14DC0378BDDE5EDD73E64F682A79D14515CF9C8D4C5A0F469AD773DF9F4BFC8F05876680DBB4A7CB546B1EE27FF10134B

其中密码计算过程：

ENCRYPT("326086233FAB8A96566796551EB1DFF8","AA30C505.20MQTTJ000121_DB6FACEF35BEBA0517712F0E4BE32936".getBytes(StandardCharsets.UTF_8)).toHexString().toUpperCase();

密码得到结果：

5EB6BBC8DE5233E632389FA3E8E133F14DC0378BDDE5EDD73E64F682A79D14515CF9C8D4C5A0F469AD773DF9F4BFC8F05876680DBB4A7CB546B1EE27FF10134B

2、type=1

假设 openId=7ED01E2A,productId=AA30C505,nodeEui=20MQTTJ000121， timestamp=1675404867,type=1,deviceSecret=326086233FAB8A96566796551EB1DFF8

那么 mqtt 连接参数如下：

MqttClientID:V2_AA30C505.20MQTTJ000121_1_1675404867

MqttUserName：7ED01E2A-AA30C505-20MQTTJ000121

MqttPassword:5EB6BBC8DE5233E632389FA3E8E133F14DC0378BDDE5EDD73E64F682A79D14515CF9C8D4C5A0F469AD773DF9F4BFC8F0725CDE013D77D74F22A59C33F8FE51622BE8CFAD9396E7811545F61CEDA9E4F9

其中密码计算过程：

ENCRYPT("326086233FAB8A96566796551EB1DFF8","AA30C505.20MQTTJ000121_DB6FACEF35BEBA0517712F0E4BE32936_1675404867".getBytes(StandardCharsets.UTF_8)).toHexString().toUpperCase();

密码得到结果：

5EB6BBC8DE5233E632389FA3E8E133F14DC0378BDDE5EDD73E64F682A79D14515CF9C8D4C5A0F469AD773DF9F4BFC8F0725CDE013D77D74F22A59C33F8FE51622BE8CFAD9396E7811545F61CEDA9E4F9

2.2.6 平台激活

设备完成上述流程后，平台会将设备标记为“在线”状态，设备后续可继续进行其他数据交互。

2.3 topic 介绍

分类	定义	topic	操作
基础通信	数据下行	/sys/${openId}/${produdtId}/${deviceId}/downlink	订阅
	认证	/sys/device/auth	发布
	设备上报固件信息	/sys/${openId}/${produdtId}/${deviceId}/firmware_report	发布
	上报固件回复	/sys/${openId}/${produdtId}/${deviceId}/firmware_report_ack	订阅
	平台查询设备固件信息	/sys/${openId}/${produdtId}/${deviceId}/ask_firmware	订阅
	下发 ota 升级指令	/sys/${openId}/${produdtId}/${deviceId}/ota_notify	订阅

topic	设备回复ota升级指令	/sys/${openId}/${produdtId}/${deviceId}/ota_notify_ack	发布
	平台获取设备配置信息	/sys/${openId}/${produdtId}/${deviceId}/ask_config	订阅
	设备上报配置信息	/sys/${openId}/${produdtId}/${deviceId}/config_report	发布
	平台下发设备配置	/sys/${openId}/${produdtId}/${deviceId}/down_config	订阅
	设备订阅的通配符	/sys/${openId}/${produdtId}/${deviceId}/+	订阅
	设备请求同步时间	/sys/${openId}/${produdtId}/${deviceId}/update_time	发布
	平台获取同步时间ack	/sys/${openId}/${produdtId}/${deviceId}/update_time_ack	订阅
透传topic	透传上报	/sys/${openId}/${produdtId}/${deviceId}/uplink	发布
	上行回复	/sys/${openId}/${produdtId}/${deviceId}/uplink_ack	订阅
	透传下行	/sys/${openId}/${produdtId}/${deviceId}/downlink	订阅
	下行回复	/sys/${openId}/${produdtId}/${deviceId}/downlink_ack	发布
物模型-JSON	属性读取	/sys/${openId}/${productId}/${deviceId}/model/property/ask	订阅
	属性上报	/sys/${openId}/${productId}/${deviceId}/model/property/report	发布
	属性上报ack	/sys/${openId}/${productId}/${deviceId}/model/property/report_ack	订阅
	属性设置	/sys/${openId}/${productId}/${deviceId}/model/property/set	订阅
	属性设置ack	/sys/${openId}/${productId}/${deviceId}/model/property/set_ack	发布
	事件上报	/sys/${openId}/${productId}/${deviceId}/model/event/${event.identifier}/report	发布
	事件上报ack	/sys/${openId}/${productId}/${deviceId}/model/event/${event.identifier}/report_ack	订阅
	服务调用	/sys/${openId}/${productId}/${deviceId}/model/service/${service.identifier}	订阅
	服务调用ack	/sys/${openId}/${productId}/${deviceId}/model/service/${service.identifier}/ack	发布
	历史数据上报	/sys/${openId}/${productId}/${deviceId}/model/history/data/report	发布
	历史数据上报ack	/sys/${openId}/${productId}/${deviceId}/model/history/data/report_ack	订阅

物模型-十六进制	上报	/sys/${openId}/${productId}/${deviceId}/model/hex/report	发布
	上报 ack	/sys/${openId}/${productId}/${deviceId}/model/hex/report_ack	订阅
	下发	/sys/${openId}/${productId}/${deviceId}/model/hex/downlink	订阅
	下发 ack	/sys/${openId}/${productId}/${deviceId}/model/hex/downlink_ack	发布
	上报全 Hex	/sys/${openId}/${productId}/${deviceId}/model/hex	发布
	上报全 Hex ack	/sys/${openId}/${productId}/${deviceId}/model/hex_ack	订阅

2.4 智能设备接入及数据的安全性

平台为了保证设备接入以及网络数据传输的安全性，采用双重密钥机制来设计接入协议，分别为入网安全密钥（DeviceSecret）和数据安全密钥（SessionKey）。数据的加密采用对称加密算法 AES128；而且数据安全密钥通过网络动态获取并定时更换，最大程度保证设备的应用数据安全。

3 接入协议数据帧格式概述

IoT 智能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、MessageId 、Version 以及 Payload（内容可自定义）。

接入协议数据的详细内容如下所示：

```
{
    "MessageId":"",
    "HeaderCtrl":1, //命令码
    "Payload":{} ,//数据部分
    "Version":"2.0"
}
```

返回 Ack 协议数据的详细内容如下所示：

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

```
"HeaderCtrl":1, //命令码
"Payload":{
    "Code":"200",   //code 返回码见表 2.3
    "Msg":"SUCCESS",
    "Data":{}  // 非必须
},
"Version":"2.0"
}
```

协议内容中的各个字段说明如表 2.1 所示。

字段	类型	说明
MessageId	String	消息 Id
HeaderCtrl	int	协议的控制命令：指令定义详见表 2.2。
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
Version	String	协议版本

批注：协议标识号为第 2 页“文档管理信息”表中的“协议标识号”

表 2.1 字段说明

Command	指令名称	说明
智能设备主动发起的 Command		
1	智能设备请求认证平台	智能设备跟 IoT 平台进行任何数据通信之前，可以先对平台的身份进行认证
4	应用数据上行（Uplink）	智能设备将应用数据上报给 IoT 平台，平台根据路由规则转发给与设备关联的应用
8	智能设备上报固件版本号	智能设备向 IoT 平台上报自己的固件版本号
14	设备配置指令	应用获取和更改当前设备的配置
16	物模型属性上报(JSON)	智能设备主动向平台上报物模型定义的属性，数据格式为 JSON
17	物模型事件上报(JSON)	智能设备主动向平台上报物模型定义的事件，数据格式为 JSON

18	物模型属性上报(HEX)	智能设备主动向平台上报物模型定义的属性，数据格式为十六进制
19	物模型事件上报(HEX)	智能设备主动向平台上报物模型定义的事件，数据格式为十六进制
26	物模型服务调用 ack(HEX)	IoT 平台通过物模型服务调用后，智能设备将应答情况反馈给平台
27	物模型历史数据上报(JSON)	智能设备主动向平台上报物模型定义的历史属性数据，数据格式为 JSON
28	物模型全 Hex 数据上报(HEX)	智能设备主动向平台上报全 Hex 的物模型数据
29	时间同步 topic	设备请求同步时间
IoT 平台主动发起的 Command		
3	更新智能设备 SessionKey	IoT 平台监测到智能设备的 SessionKey 过期后,需要向设备发送最新的 SessionKey
7	应用数据下行（Downlink）	应用数据下行。客户的应用服务器请求 IoT 平台的接口下发应用数据给智能设备，平台将数据打包后下发给智能设备
9	IoT 平台推送固件更新消息	IoT 平台主动向智能设备推送固件更新的消息。
13	平台查询设备的固件版本号	IoT 平台主动查询智能设备的固件版本号，设备收到此指令后需要使用指令 8 来向 IoT 平台上报自己的固件版本号
20	物模型属性设置(JSON)	IoT 平台通过指令 20,设置设备在物模型定义中所定义的属性，数据格式为 JSON
21	物模型服务调用(JSON)	IoT 平台通过指令 21,调用设备在物模型定义中所定义的服务，数据格式 JSON
23	物模型属性查询(JSON)	IoT 平台通过指令 23，查询智能设备的属性值，数据格式 JSON
22	物模型属性设置(HEX)	IoT 平台通过指令 22,设置设备在物模型定义中所定义的属性，数据格式为十六进制

24	物模型服务调用(HEX)	IoT 平台通过指令 24,调用设备在物模型定义中所定义的服务,数据格式为十六进制
25	物模型属性查询(JEX)	IoT 平台通过指令 25，查询智能设备的属性值，数据格式为十六进制

表 2.2 数据帧 Command 指令说明

字段	
200	操作成功
4001	操作失败

表 2.3 ack 返回 code 码表

4 智能设备与 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 2.2 智能设备接入流程示例的要求接入,并确保设备在线。下面主要描述了：透传设备数据交互、物模型设备数据交互、OTA 交互等流程。更多详细信息，请查看对应章节。

透传设备数据交互

在认证入网之后，我们就可以进行数据交互了。设备使用 HeaderCtrl：4 进行数据上报。同样服务器会使用 HeaderCtrl：4 通过**上行回复 topic** 进行应答。设备订阅任意 topic 即可,推荐使用上行回复 topic。具体参考 2.3-topic 介绍。

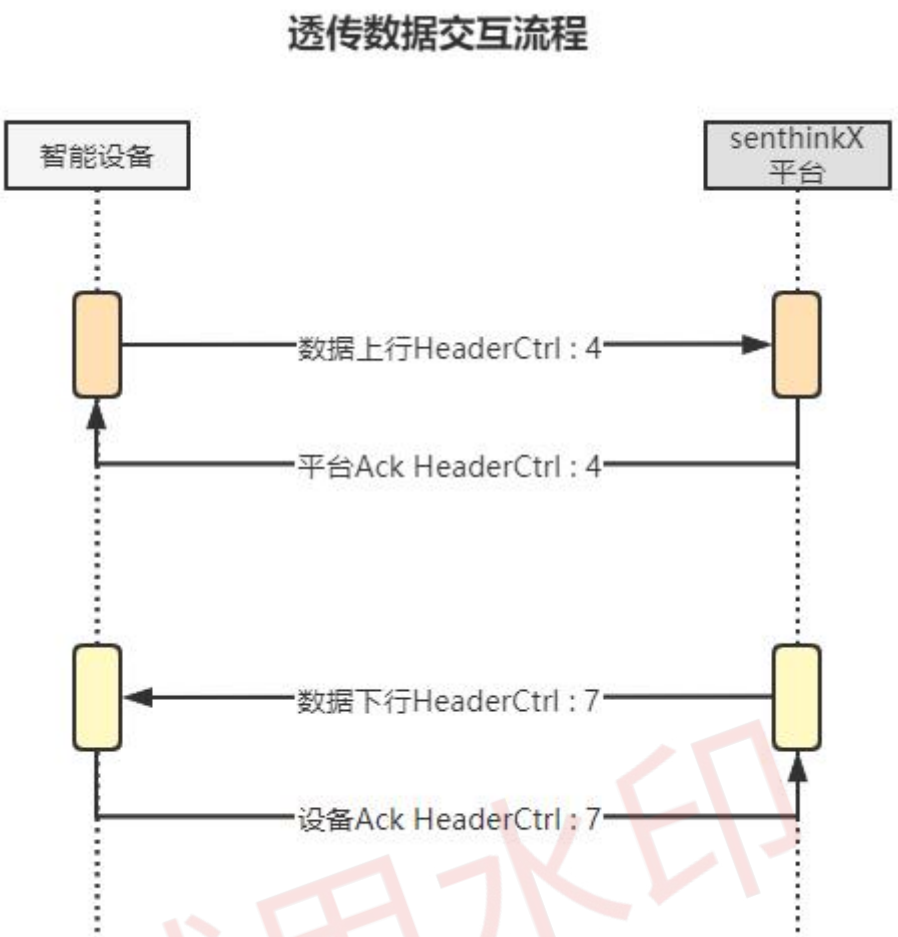


图 4.2 智能设备与 X 平台透传数据交互流程

物模型设备数据交互

物模型交互分 2 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。

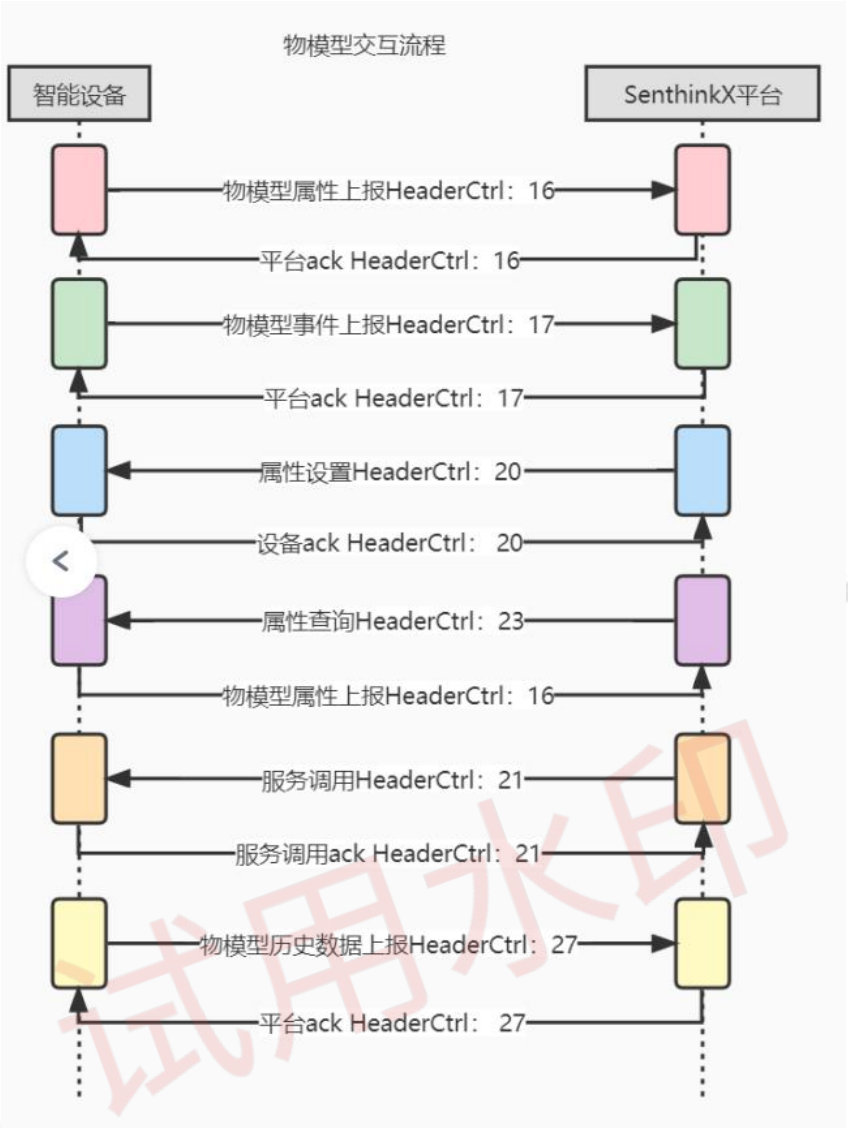


图 4.3 智能设备与 X 平台物模型数据交互流程

ota 升级

我们建议， 设备入网之后上报设备固件信息，或者定期上报设备固件信息。在后台创建设备升级任务之后，设备通过上报版本号触发平台下发升级任务。设备收到升级升级信息之后，进行升级。 升级结束完成，重新进行认证入网，固件上报流程。

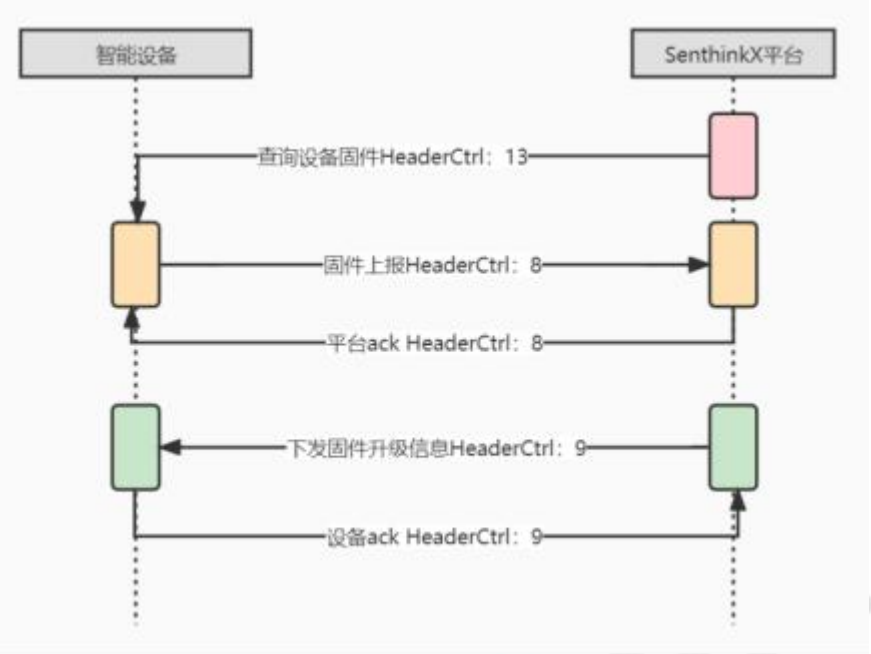


图 4.4 智能设备与 X 平台 OTA 交互流程

4.1 智能设备认证 IoT 平台（Auth）

设备上报认证的通用 Topic：/sys/device/auth

认证统一使用产品的账号密码进行认证

MqttClientID：${ProductID}-${NodeEui(设备 ID)}

MqttUserName：AUTH-${OpenID}-${ProductID}

MqttPassword：${ProductKey}

备注：一型一密免注册和一型一密预注册需要，一机一密设备可以自行选择是否认证。此外，加密设备，无论是免注册设备预注册设备，都需走认证流程以获取 sessionID。

智能设备在正式发布其他数据前，建议先向平台发送认证请求以验证平台的身份，在信任平台的身份后才进行数据交互。智能设备认证 IoT 平台数据帧使用的 Command 为 1，数据格式和内容如下所示：

```
{
    "MessageId":1231,
    "HeaderCtrl":1,//命令码
    "Payload":{
            "OpenID":"E0C040B1",
```

```
            "ProductID":"A4BC6002",
            "NodeEui":"A4BC6002",
            "Type":0, //0:一型一密免注册，1：一型一密预注册
            },
    "Version":"2.0"
}
```

设备向 IoT 平台发起认证请求的数据帧中包含了 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
NodeEui	硬件设备所属设备的唯一标识 ID，由硬件自行分配，支持英文字母、数字长度限制 4~32 个字符
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}/${deviceId}/downlink：

IoT 平台使用同样的 Command（1）对设备进行响应，平台返回设备的 Code 和 DeviceSecret。

响应设备的数据帧格式以及内容如下表所示：

```
{
    "MessageId":1231,
    "HeaderCtrl":1,//命令码
    "Payload":{
        "Code":0,
        "DeviceSecret":"FE448731"
    },
    "Version":"2.0"
}
```

如若是加密设备，会额外返回 sessionId 和 nonce，以方便设备计算 sessionKey:

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

```
    "HeaderCtrl":1, //命令码
    "Payload":{
        "Code":0,
        "DeviceSecret":"FE448731",
        "SessionID":"3788433",
        "Nonce":4
    },
    "Version":"2.0"
}
```


设备认证回复信息中的 Code 属性说明：

对于智能设备认证 IoT 平台指令（Auth），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	认证请求处理成功：表示 IoT 平台成功处理请求并返回签名后的 deviceSecret 值。
1	认证请求处理失败：IoT 平台系统繁忙，暂时无法处理认证请求。**建议智能设备重试**。
2	认证请求处理失败：认证所使用的产品未注册，无法处理认证请求。IoT 平台会将 Random 值原样返回给智能设备。**建议检查 OpenID 和 ProductID 是否正确**。
4	认证请求处理失败：**该设备已被禁用或者设备 nodeeui 号非法**
5	参数 type 类型错误，请检查 type 类型和平台产品 type 定义一致
6	设备版本号与上送的协议版本号不一致

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，使用认证结果返回的 deviceSecret,根据 2.2.5 设备连接的计算方法,计算连接接 mqtt broker 的 MqttClientID,MqttUsername 和 MqttPassword，连接 mqtt broker，之后就可以进行数据上报等操作了。

如果智能设备发送了认证请求，等待超时后没有收到平台的任何回复，则有可能是网络问题导致数据丢失，建议检查网络并重试。

4.2 透传产品数据交互

4.2.1 应用数据上行（Uplink）

设备发送特定上行数据 Topic：/sys/${openId}/${productId}/${deviceId}/uplink

智能设备成功入网后，可以进行应用数据的交互，使用 Command 为 4 的指令来上行用户的应用数据。上行数据帧的格式和内容如下所示：

（1）未加密传输格式

```
{
    "MessageId" :1231,
    "HeaderCtrl":4, //命令码
    "Payload":{
        "deviceId":"10000001",
        "power":99
    },
    "Version":"2.0"
}
```

（2）加密传输格式

```
{
    "MessageId":1231,
    "HeaderCtrl":4, //命令码
    "Payload":"CB96395F5065FD121D88B1593EFE",    //加密后 16 进制传输
    "Version":"2.0"
}
```

IoT 平台会将设备的上行应用数据按照设定好的路由规则，分发给上层应用服务器。如果使用数据加密模式，设备必须使用 SessionKey 对中的应用数据有效载荷 Payload 进行加密之后再进行网络传输；非加密模式则不需要加密 Payload。

加密方法如下描述（伪代码）：

加密方式：AES128_ECB_PKCS5Padding

Payload= ENCRYPT(SessionKey, Content);

假设：

SessionKey：A9 AB 8E 82 D2 D4 A6 0E 67 C9 9D 3A 44 6D 60 88

Payload（json）：{"deviceId":"10000001","power":99}

加密结果（16 进制）：CB96395F5065FD121D88B1593EFE

解密方法如下描述（伪代码）：

解密方式：AES128_ECB_PKCS5Padding

Content= DECRYPT(SessionKey , Payload);

IoT 平台收到设备上报之后会通过如下 topic 对设备上报进行应答，

uplink 应答 topic：/sys/${openId}/${produdtId}/${deviceId}/uplink_ack

应答数据结构如下

```
{
    "MessageId":123123,
    "HeaderCtrl":4, // 应答的 HeaderCtrl 和 Uplink 中的 HeaderCtrl 相同
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS"
    },
    "Version":"2.0"
}
```

4.2.2 应用数据下行（Downlink）

下行发送特定的 Topic：/sys/${openId}/${productId}/${deviceId}/downlink

客户的应用服务器可以通过 IoT 平台提供的标准接口下发应用数据给智能设备，平台会将应用数据进行打包然后下发给指定的设备。下行数据帧使用的 Command 为 7；数据帧格式和内容如下所示：

```
{
  "MessageId":1231,
    "HeaderCtrl":7, //命令码
```

```
    "Payload":{
        "data":"13234234....."//下行数据
    },
    "Version":"2.0"
}
```

如果应用服务器指定了加密传输，平台会将 Payload 加密后传输给智能设备。加解密方式参考 4.3.1 应用数据上行（Uplink）

设备收到下行数据之后，应当通过如下 topic 进行应答。

设备端应答 topic: /sys/${openId}/${produdtId}/${deviceId}/downlink_ack

数据结构示例

```
{
    "MessageId":1231,
    "HeaderCtrl": 7,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data": {}  // 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.3 物模型产品数据交互

4.3.1 物模型 JSON 格式产品数据交互

前提：后台创建产品的时候选择非透传，消息格式选择为 JSON，并定义好产品物模型，后面属性上报、事件上报和服务调用都是基于定义的物模型。

4.3.1.1 属性上报

topic: /sys/${openId}/${productId}/${deviceId}/model/property/report

智能设备根据后台物模型定义的属性，将具体属性数据以 JSON 格式上报给平台。上行数据帧使用的 Command 为 16，数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":16,//命令码
    "Payload":{
        "temp":100, // 属性
        "humi":20// 属性
    },
    "Version":"2.0"
}
```

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}/${deviceId}/model/property/report_ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":16,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data": {}// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.3.1.2 事件上报

topic:/sys/${openId}/${productId}/${deviceId}/model/event/${event.identifier}/report

智能设备根据后台物模型定义的事件，将具体事件数据以 JSON 格式上报给平台。上行数据帧使用的 Command 为 17。

注意：${event.identifier}为定义产品物模型的时候，定义的事件的标识符。数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":17,//命令码
    "Payload":{
        "temp":100, // 输出参数
        "humi":20// 输出参数
    },
    "Version":"2.0"
}
```

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}/${deviceId}/model/event/${event.identifier}/report_ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":17,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data": {}// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.3.1.3 服务调用

topic:/sys/${openId}/${productId}/${deviceId}/model/service/${service.identifier}

平台根据后台物模型定义的服务，将服务指令以 JSON 格式下发给智能设备。智能设备根据下发指令，执行对应的服务操作。下行数据帧使用的 Command 为 21。

注意: ${service.identifier} 为定义产品的时候，定义的服务的标识符。数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":21, //命令码
    "Payload":{
        "serviceIdentify":"switch",// 服务标识
        "params":{
                "on":1// 输入参数
                }
    },
    "Version":"2.0"
}
```

设备收到服务调用参数之后应当通过如下 topic 进行应答 ack。

topic:/sys/${openId}/${productId}/${deviceId}/model/service/${service.identifier}/ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":21,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data":{}// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
```

```
    "Version":"2.0"
}
```

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。

4.3.1.4 历史数据上报

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

智能设备根据后台物模型定义的属性和事件，将具体属性数据以 JSON 格式上报给平台。上行数据帧使用的 Command 为 27，其中，时间戳为毫秒，数据内容格式如下：

```
{
    "MessageId": 1231,
    "HeaderCtrl":27,//命令码
    "Payload":{
        "params": [
            {
              "properties": [
                {
                  "Power": {
                    "value": "on",
                    "time": 1524448722000
                  },
                  "WF": {
                    "value": "3",
                    "time": 1524448722000
                  }
                },
                {
                  "Power": {
                    "value": "on",
```

```
            "time": 1524448722000
          },
          "WF": {
            "value": "3",
            "time": 1524448722000
          }
        }
      ],
      "events": [
        {
          "alarmEvent": {
            "value": {
              "Power": "on",
              "WF": "2"
            },
            "time": 1524448722000
          },
          "alertEvent": {
            "value": {
              "Power": "off",
              "WF": "3"
            },
            "time": 1524448722000
          }
        }
      ]
    },
    {
      "properties": [
        {
```

```
        "Power": {
          "value": "on",
          "time": 1524448722000
        },
        "WF": {
          "value": "3",
          "time": 1524448722000
        }
      }
    ],
    "events": [
      {
        "alarmEvent": {
          "value": {
            "Power": "on",
            "WF": "2"
          },
          "time": 1524448722000
        },
        "alertEvent": {
          "value": {
            "Power": "off",
            "WF": "3"
          },
          "time": 1524448722000
        }
      }
    ]
  }
]
```

```
    },
    "Version":"2.0"
}
```

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}/${deviceId}/model/history/data/report_ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":27,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data": {}// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.3.1.5 属性设置

topic: /sys/${openId}/${productId}/${deviceId}/model/property/set

平台根据智能设备已上报的属性，可将对应的属性设置指令以 JSON 格式下发给智能设备。智能设备根据下发指令，执行属性设置操作。下行数据帧使用的 Command 为 20，数据内容格式如下：

```
{
    "MessageId":192155183,
    "HeaderCtrl":20,
    "Payload":{
    "params":[{
        "identify":"temp",// 属性标识
        "value": "100"// 属性值
            }]},
```

```
        "Version":"2.0"
}
```

智能设备收到属性设置指令之后会通过属性设置上报 topic 进行应答 ack。

topic: /sys/${openId}/${productId}/${deviceId}/model/property/set_ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":20,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data":{
          "temp":"100"
        }// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

设置失败上送内容：

```
{
    "MessageId":1231,
    "HeaderCtrl":20,
    "Payload":{
        "Code":"4001",
        "Msg":"Failed",
        "Data":{}  // 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.3.1.6 属性查询

topic: /sys/${openId}/${productId}/${deviceId}/model/property/ask

平台根据智能设备已上报的属性，可将对应的属性查询指令以 JSON 格式下发给智能设备。智能设备根据下发指令，执行对应的属性查询操作。下行数据帧使用的 Command 为 23，数据内容格式如下：

```
{
    "MessageId":192155183,
    "HeaderCtrl":23,
    "Payload":{
        "params": ["temp"]
    },
    "Version":"2.0"
}
```

智能设备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}/${deviceId}/model/property/report

数据内容格式如下：

```
{
    "MessageId": 192155183,
    "HeaderCtrl":16,
    "Payload":{
        "temp":100//属性
    },
    "Version":"2.0"
}
```

4.3.2 物模型 HEX 格式产品数据交互

使用前请参考 https://help.senthink.com/物模型解析脚本说明。

前提：产品定义时选择物模型，消息格式选择 HEX 格式，并且已经创建了产品对应的物模型和解析脚本。

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 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 payload 部分是经过解析脚本编码之后的 16 进制格式数据。

4.3.2.1 物模型 16 进制上报

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

智能设备根据平台物模型定义，将具体数据以十六进制格式上报给平台。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
18	属性上报
19	事件上报
26	服务调用 ack

注意：payload 部分的编码规则，需要结合自己的解析脚本。数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":19, //18 属性上报, 19 事件上报
    "Payload": "00f0ff013FA00000",
    "Version":"2.0"
  }
```

服务端收到上报之后会通过如下 topic 进行应答 ack

topic: /sys/${openId}/${productId}/${deviceId}/model/hex/report_ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":18,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
```

```
        "Data":{}// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.3.2.2 物模型 16 进制下发

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

平台根据后台的物模型定义，将指令以十六进制格式下发给智能设备。智能设备根据下发指令，执行相应操作。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
22	属性设置
24	服务调用
25	属性查询

注意：payload 部分的编码规则，需要结合自己的解析脚本。数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":22, //22 属性设置, 24 服务调用, 25 属性查询
    "Payload":"0101002d0142f6e76d",
    "Version":"2.0"
  }
```

设备收到下发指令会通过如下 topic 应答服务端

topic:/sys/${openId}/${productId}/${deviceId}/model/hex/downlink_ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":22,
    "Payload":{
        "Code":"200",
```

```
        "Msg":"SUCCESS",
        "Data":{}// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.3.2.3 物模型全 16 进制数据上报

topic:/sys/${productId}/${deviceId}/model/hex

智能设备根据平台物模型定义，将具体数据以全部十六进制格式的方式上报给平台。

注意:对于加密设备来说，全 hex 上送无需加密;全 Hex 方式的数据，设备除了上报应用数据以外，还需要附带属性上报、事件上报消息区分的字段，具体的编码规则，需要结合自己的解析脚本。数据内容格式如下：

```
"00f0ff013FA00000"
```

服务端收到上报之后会通过如下 topic 进行应答 ack

topic: /sys/${productId}/${deviceId}/model/hex_ack

数据内容格式如下：

```
{
    "MessageId":1231,
    "HeaderCtrl":28,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data": {}  // 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

4.4 设备平台主动更新 SessionKey（加密模式）

平台下发 Topic：/sys/${openId}/${productId}/${deviceId}/downlink

特别注意：如果智能设备所属的产品类型是加密产品，则表示开启 SessionKey 检测的功能；平台会设备连接时候检测 SessionKey 是否过期，如果检测到智能设备的 SessionKey 过期（默认 48 小时过期），平台会主动向智能设备推送新的 SessionKey。

更新 SessionKey 数据帧使用的 Command 为 **3**；数据帧格式和内容如下：

```
{
  "MessageId":1231,
        "HeaderCtrl":3,//命令码
        "Payload":{
            "SessionID":"4655434B",//新的 sessionId
            "Nonce":128
        },
        "Version":"2.0"
    }
```

智能设备收到响应数据帧后，**设备需要使用入网安全密钥 ProductKey，对 SessionID 和 Nonce 拼接的 byte 数组数据安全密钥 SessionKey**，然后下次上报数据时使用新的 SessionKey 对数据进行加密。

计算方法如下描述（伪代码）：

加密方式：AES128_ECB_PKCS5Padding

```
SessionKey = ENCRYPT( ProductKey, SessionID+nonce.toHex);
```

4.5 设备配置指令(Setting)

4.5.1 获取配置

应用服务器，可以通过接口下发获取设备配置信息，平台收到请求后，IoT 平台收到请求后，会下发命令到智能设备，下行数据帧使用的 Command 为 14；数据帧格式和内容如下所示：

Topic：/sys/${openId}/${productId}/${deviceId}/ask_config

```
{
```

```
    "HeaderCtrl":14,
    "Payload":{},
    "MessageId":1231,
    "Version":"2.0"
}
```

4.5.2 设备配置上报

只能设备收到平台下发请求配置信息之后，需要通过配置上报的 topic 进行配置上报，数据格式和 topic 如下。

Topic：/sys/${openId}/${productId}/${deviceId}/config_report

```
{
    "MessageId":1231,
    "HeaderCtrl":14,
    "Payload":{
        "Settings":{
        "Host":"10.200.0.135",
        "Port":7200,
        .....
        }
    },
    "Version":"2.0"
}
```

4.5.3 服务端下发配置

客户可以通过 IOT 平台给设备下发配置信息，配置信息的内容下发的 topic 和内容格式如下，setting 为配置信息，为 key，value 结构

Topic：/sys/${openId}/${productId}/${deviceId}/down_config

```
{
```

```
        "MessageId":294592440,
        "HeaderCtrl":14,
        "Payload":{
            "Settings":"{
                "data":"1231"......
        }"
          },
        "Version":"2.0"
    }
```

智能设备收到配置后，应当通过 4.5.2 上报配置来作为应答。

4.6 同步设备时间

注：json 类型产品和 hex 类型产品若需时间同步功能都需上报 JSON 格式的请求体对于加密设备来说，时间同步上送请求体无需加密。

设备请求同步时间 Topic:/sys/${openId}/${produdtId}/${deviceId}/update_time

```
    {
  "MessageId":1231,
        "HeaderCtrl":29,//命令码
        "Payload":"",
        "Version":"2.0"
    }
```

平台收到后，下发 ack 到设备。

平台 ackTopic:/sys/${openId}/${produdtId}/${deviceId}/update_time_ack

平台下发当前时间到设备，设备用来校准时间，使用的 Command 为 29；数据帧格式和内容如下：

```
    {
        "MessageId":123123,
        "HeaderCtrl":29,
        "Payload":{
            "Code":"200",
            "Msg":"SUCCESS",
```

```
        "Data": {
            "timestamp":123213122332,//当前时间戳，到 ms
            "timezone":"Asia/Shanghai"//当前时区
        }
    },
    "Version":"2.0"
}
```

5 智能设备 OTA 升级

IoT 平台为智能硬件设备提供远程在线升级（OTA 升级）的功能。如果 IoT 平台上有可用于更新的固件，平台会将可用更新的固件信息推送给智能设备。**如果智能设备不需要 OTA 升级功能，则第 5 章【智能设备 OTA 升级】的内容可以忽略。MQTT 目前仅支持 HTTP 升级方式。**

5.1 智能设备上报固件版本号

topic: /sys/${openId}/${produdtId}/${deviceId}/firmware_report

使用的 Command 为 8；数据帧格式和内容如下：

```
{
    "MessageId":1231,//消息 id
    "HeaderCtrl":8, //命令码
    "Payload":{
        "HardVer":"v1.0.1",//硬件版本
        "SoftVer":"v2.1.3",//软件版本
        "Type":2, //升级方式：2-http
        "Module":"module1"//模块标识，默认“default”模块可以不上报此参数；默认“default”模块代表整个设备的主版本号
    },
    "Version":"2.0"
```

```
}
```

服务器收到固件上报之后，会通过固件上报应答的 topic 下发应答。topic 和数据结构如下。

topic: /sys/${openId}/${produdtId}/${deviceId}/firmware_report_ack

数据结构:

```
{
    "MessageId":123123,
    "HeaderCtrl":8,
    "Payload":{
        "Code":"200",
        "Msg":"SUCCESS",
        "Data":{}// 如果有输出参数，则对应 Data 字段， 若无，该字段不返回。
    },
    "Version":"2.0"
}
```

在上报固件版本号的过程中，可能会由于网络原因没有上报到平台或者平台的响应没有到达设备端；此情况下，智能设备应该重新上报，至少尝试重发 1 次，建议重发 2~3 次，重发的时间间隔由设备根据自己的实际情况决定。

上报版本号后，检查当前是否有未完成的 OTA 升级任务，如果有的话将收到版本与之前版本比对，如果相同则上报升级成功消息到应用服务器，如果不同，则上报升级失败消息到应用服务器。

5.2 IoT 平台查询智能设备的固件版本号

IoT 平台可以通过 Command 为 13 的指令向智能设备查询设备的固件版本号，智能设备收到此条指令后，需要使用 Command 为 8 的指令向平台上报自己的固件版本号（如 5.1 智能设备上报固件版本号的章节所描述）。

topic: /sys/${openId}/${productId}/${deviceId}/ask_firmware

IoT 平台向智能设备查询固件版本号的数据帧格式和内容如下所示：

```
{
    "MessageId":1231, //消息 id
    "HeaderCtrl":13, //命令码
```

```
    "Payload":{},
    "Version":"2.0"
}
```

5.3 IoT 平台向智能设备推送固件更新消息

如果 IoT 平台上有可用于升级的固件版本，平台会使用 Command 为 9 的指令向智能设备推送固件更新消息。推送的 topic 和消息数据帧格式和内容如下所示:

topic: /sys/${productId}/${deviceId}/ota_notify

```
{
    "MessageId":1231, //消息 id
    "HeaderCtrl":9, //命令码
    "Payload":{
        "Type":2, //升级方式：2-http
        ... //具体内容和格式如 5.3.1 章节的描述
    },
    "Version":"2.0"
}
```

推送的数据帧中包含了 Payload 域，携带具体更新固件以及升级服务器等信息。

IoT 平台会根据智能设备上报固件版本号时所设置的 OTA 升级方式（Type）字段来向智能设备推送具体信息。具体内容和格式如 5.3.1 章节描述。

5.3.1 智能设备的固件升级方式为 HTTP

当智能设备上报固件版本号时所设置的 Type 为 2，即设备使用 HTTP 方式升级固件，IoT 平台会向智能设备推送固件信息和 HTTP 固件下载链接的信息，推送的数据帧中 Payload 的格式和内容如下所示：

```
    "Payload":{
        "Type":2,//升级方式：2-http
        "Url":"http://ota.iot.senthink.com/02022117",
```

```
"SoftVer":"v2.1.4",
"SoftSize":123345435, //固件大小
"MD5":232435456, //MD5 校验值
"DownID":"E0C040B1",
"firmwareName":"BOX02000_100_103_C_0_0",//升级固件文件名称，如果固件名称设置中文，需要硬件解码
"Module":"module1"//模块标识，模块为默认模块“defalut”时，平台不下发 Module 参数
}
```

Payload 域中包含的内容及说明如下所示：

Payload	说明
Type	这里的 Type 对应智能设备上报固件版本号时所设置的 Type。
Url	IoT 平台上可用于更新的固件的 HTTP 下载链接：格式为字符串。 例如：HTTP 下载固件的链接为 http://ota.iot.senthink.com/02022117
SoftVer	IoT 平台上可用于更新的固件版本号，格式为字符串。
SoftSize	IoT 平台上可用于更新的固件文件的大小，单位为字节。
MD5	IoT 平台上可用于更新的固件文件的 MD5 校验值。
DownID	可用更新的固件的下载 ID(DownID):智能设备使用 HTTP 方式下载固件文件的时候,需要带上 DownID（HTTP 下载固件流程会介绍）。
Module	升级的模块标识,模块为默认模块“default”时，平台不下发 Module 参数。

综合 5.3.1 章节的内容描述，智能设备在收到 IoT 平台推送的固件更新消息后，需要**首先使用**同样的 Command（9）向平台回复确认数据帧。设备可以使用如下两个 topic 中任意一个回复升级任务，topic 如下

topic:/sys/{openId}/{productId}/{deviceId}/ota_notify_ack。

回复的确认数据帧格式和内容如下所示：

```
{
"MessageId":1231, //消息 id
"HeaderCtrl":9, //命令码
"Payload":{
```

"DownID": "33367341"
},
"Version":"2.0"
}

在回复平台的确认数据帧中需要带上 IoT 平台推送的 DownID。

特别注意：在成功向平台回复确认数据帧后，智能设备需要将自己实际的固件版本号与平台推送的固件版本号进行对比（ASCII 字符串对比），如果一样则不需要对目标固件进行下载更新，不一样则需要下载并更新。

6 智能设备与 IoT 平台交互注意事项

6.1 安全性及健壮性考虑

为了保证接入以及数据的安全性，硬件设备开发人员应该严格遵从协议规范进行开发。