设备接入

用户指南

文档版本 1.47

发布日期 2022-06-25

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 概述 1

2 设备接入 3

2.1 原生协议接入 3
2.1.1 设备鉴权 3
2.1.2 MQTT/MQTTS 协议接入 6
2.1.2.1 使用前必读 6
2.1.2.2 设备日志收集 7
2.1.2.2.1 平台下发日志收集通知 7
2.1.2.2.2 设备上报日志内容 8
2.1.2.3 设备信息上报 10
2.1.2.3.1 设备信息上报 10
2.1.2.4 设备时间同步 11
2.1.2.4.1 设备时间同步请求 11
2.1.2.4.2 设备时间同步响应 12
2.1.2.5 文件上传/下载管理 13
2.1.2.5.1 设备上报获取文件上传 URL 请求 13
2.1.2.5.2 平台下发文件上传临时 URL 15
2.1.2.5.3 设备上报文件上传结果 16
2.1.2.5.4 设备上报获取文件下载 URL 请求 17
2.1.2.5.5 平台下发文件下载临时 URL 19
2.1.2.5.6 设备上报文件下载结果 20
2.1.2.6 软固件升级 21
2.1.2.6.1 平台下发获取版本信息通知 21
2.1.2.6.2 设备上报软固件版本 22
2.1.2.6.3 平台下发升级通知 24
2.1.2.6.4 设备上报升级状态 26
2.1.2.7 网关与子设备管理 28
2.1.2.7.1 平台通知网关子设备新增 28
2.1.2.7.2 平台通知网关子设备删除 30
2.1.2.7.3 网关同步子设备列表 32
2.1.2.7.4 网关更新子设备状态 33
2.1.2.7.5 网关上报获取平台产品信息消息 34
2.1.2.7.6 平台下发产品信息响应消息 35

2.1.2.7.7 平台下发启动子设备扫描通知 37
2.1.2.7.8 网关上报子设备列表信息 38
2.1.2.7.9 网关新增子设备请求 40
2.1.2.7.10 网关新增子设备请求响应 42
2.1.2.7.11 网关删除子设备请求 44
2.1.2.7.12 网关删除子设备请求响应 45
2.1.2.7.13 网关更新子设备请求 47
2.1.2.7.14 网关更新子设备请求响应 49
2.1.2.8 设备属性 51
2.1.2.8.1 设备属性上报 51
2.1.2.8.2 网关批量设备属性上报 52
2.1.2.8.3 平台设置设备属性 54
2.1.2.8.4 平台查询设备属性 56
2.1.2.8.5 设备侧获取平台的设备影子数据 57
2.1.2.9 设备消息 59
2.1.2.9.1 设备消息上报 59
2.1.2.9.2 平台消息下发 60
2.1.2.10 设备命令 61
2.1.2.10.1 平台命令下发 61
2.1.2.11 设备连接鉴权 63
2.1.2.12 Topic 定义 64
2.1.2.13 通信方式概述 66
2.1.3 HTTPS 协议接入 67
2.1.3.1 使用 https 协议接入 67
2.1.3.2 API 概览 70
2.1.3.3 设备鉴权 70
2.1.3.3.1 设备鉴权 70
2.1.3.4 设备消息上报 72
2.1.3.4.1 设备消息上报 72
2.1.3.5 设备属性上报 74
2.1.3.5.1 设备属性上报 74
2.1.3.5.2 网关上报子设备属性 76
3 设备管理 80
3.1 创建产品 80
3.2 产品模型 81
3.3 注册设备 83
3.3.1 注册单个设备 83
3.3.2 批量注册设备 85
3.3.3 注册 X.509 证书认证的设备 87
3.4 管理设备 91
3.5 群组与标签 94
3.6 设备影子 96

3.7 固件升级 101
3.8 软件升级 108
3.8.1 升级指导 108
3.8.2 设备侧适配开发指导 114
3.8.3 PCP 协议介绍 132
3.9 文件上传 138
3.10 网关与子设备 140
4 消息通信 144
4.1 概述 144
4.2 自定义 Topic 通信 145
4.3 数据上报 147
4.4 编解码插件 148
4.5 命令下发 150
4.5.1 命令下发机制 151
4.5.2 LwM2M/CoAP 设备命令下发 151
4.5.3 MQTT 设备命令/属性/消息下发 157
4.6 订阅推送 163
4.6.1 订阅推送方式概述 163
4.6.2 使用 HTTP/HTTPS 订阅推送 164
4.6.3 使用 AMQP 订阅推送 170
4.6.3.1 AMQP 订阅推送 170
4.6.3.2 配置 AMQP 服务端订阅 172
4.6.3.3 AMQP 客户端接入说明 174
4.6.3.4 Java SDK 接入示例 176
4.6.3.5 Node.js SDK 接入示例 182
4.6.3.6 C# SDK 接入示例 182
4.6.3.7 Android SDK 接入示例 186
5 规则引擎 187
5.1 规则引擎介绍 187
5.2 数据转发 188
5.3 设备联动 192
5.4 SQL 语句 195
5.5 连通性测试 198
6 监控运维 200
6.1 设备消息跟踪 200
6.2 查看报表 202
6.3 告警管理 203
6.4 查看审计日志 204
6.5 查看运行日志 206
7 存储管理 211
8 资源空间 214

9 IoTDA 实例 216
9.1 实例介绍 216
9.2 购买实例 217
9.3 实例管理 220
9.3.1 实例规格升配 220
9.3.2 切换实例类型 221
9.3.3 实例续费/实例退订 222
9.3.4 修改企业版实例信息 222
9.3.5 设置企业版实例公网访问 223
9.3.6 设置企业版实例专线访问 223
10 权限管理 225
10.1 创建 IAM 用户并授权使用 IoTDA 225
10.2 创建 IoTDA 自定义策略 226
11 设备发放 229
11.1 从这里开始 229
11.2 证书 230
11.2.1 制作 CA 证书 232
11.2.2 上传 CA 证书 233
11.2.3 验证 CA 证书 234
11.2.4 更新 CA 证书 235
11.2.5 删除 CA 证书 236
11.3 策略 236
11.3.1 自定义策略 236
11.3.1.1 创建自定义策略函数 236
11.3.1.2 添加自定义策略实例 244
11.3.2 证书策略 245
11.3.3 静态策略 246
11.4 设备 247
11.4.1 注册设备 247
11.4.2 注册组 252
11.5 设备数据迁移 252

1 概述

物联网平台提供海量设备的接入和管理能力，配合华为云其他产品同时使用，帮助快速构筑物联网应用，简化海量设备管理复杂性，节省人工操作，提升管理效率。使用设备接入控制台，可以实现对产品的创建、开发、调试，设备的注册、管理、鉴权、软固件升级。在设备接入控制台，可以创建规则引擎，满足用户实现设备联动和数据转发的需求；还可以存储产品和设备数据及生成相应统计报表，方便用户监控设备的各种状态。

功能	简介
产品	某一类具有相同能力或特征的设备的集合称为一款产品。您可以基于控制台快速进行产品模型和插件的开发，同时提供在线调试、自定义Topic等多种能力，端到端指引物联网开发，帮助开发者提升集成开发效率、缩短物联网解决方案建设周期。
产品模型	又称Profile，用于定义一款接入设备所具备的属性（如颜色、大小、采集的数据、可识别的指令或者设备上报的事件等信息）。产品模型可以在设备接入控制台进行无码化开发。
设备	归属于某个产品下的设备实体，每个设备具有一个唯一的标识码。设备可以是直连物联网平台的设备，也可以是代理子设备连接物联网平台的网关。
设备鉴权	设备接入物联网平台时，需要对接入平台的设备进行鉴权认证。目前华为物联网平台支持设备使用密钥和X.509两种方式进行鉴权认证。待平台验证通过，设备成功连接到物联网平台后，就可以进行数据通信。
群组与标签	群组是一系列设备的集合，用户可以对资源空间下所有设备，根据区域、类型等不同规则进行分类建立群组，以便处理对海量设备的批量管理和操作。 物联网平台支持定义不同的标签，并对设备打标签。
软固件升级	用户可以通过OTA的方式对LwM2M协议和MQTT协议的设备进行软固件升级。
设备影子	设备影子是一个JSON文件，用于存储设备的在线状态、设备最近一次上报的设备属性、应用服务器期望下发的配置。每个设备有且只有一个设备影子，设备可以获取和设置设备影子以此来同步状态，这个同步可以是影子同步给设备，也可以是设备同步给影子。

功能	简介
网关与子设备	物联网平台支持设备直连，也支持设备挂载在网关上，作为网关的子设备，由网关直连，通过网关进行数据转发。
规则引擎	用户可以在物联网平台上对接入平台的设备设定相应的规则，在条件满足所设定的规则后，平台会触发相应的动作来满足用户需求。包含设备联动和数据转发两种类型。
存储管理	设备接入控制台集成了IoT数据分析的存储管理服务，用户可以免费存储设备上报到平台的历史数据。
监控运维	提供查看统计报表、在线调试、消息跟踪、当前告警、运行日志等监控运维功能。用户可以使用这些功能，监控设备运行状态、设备消息通讯、用户操作，快速追查定位故障，保障设备的可靠性及安全性。
资源空间	可以理解为在物联网平台中为您的业务划分的一个资源空间，您在平台中创建的资源（如产品、设备等）都需要归属到某个资源空间，您可以基于资源空间实现多业务应用的分域管理。
IoTDA实例	为满足物联网不同设备规模的企业客户诉求，设备接入服务提供基础版（原共享实例）、标准版（标准实例）、企业版（专享实例）三种产品形态。您可以可以根据企业业务场景、设备规模和数据采集频率，购买最合适的实例类型和实例规格。
数据上报	当设备完成和物联网平台对接后，一旦设备上电，设备基于在设备定义上的业务逻辑进行数据采集和上报，可以是基于周期或者事件触发。
订阅推送	订阅：是指应用服务器通过调用物联网平台的API接口，向平台获取发生变更的设备业务信息（如设备生命周期管理、设备数据上报、设备消息状态、设备状态等）和管理信息（软固件升级状态和升级结果）。 推送：是指订阅成功后，物联网平台根据应用服务器订阅的数据类型，将对应的变更信息推送给指定的URL地址或AMQP消息队列。
命令下发	为能有效地对设备进行管理，设备的产品模型中定义了物联网平台可向设备下发的命令，应用服务器可以调用物联网平台开放的API接口向设备下发命令，以实现对设备的远程控制。

使用限制

为保证良好的显示效果和易用性体验，请使用兼容性良好的浏览器，对于浏览器的要求如下：

浏览器类型	版本要求	分辨率
Microsoft Edge	支持和测试最新的3个稳定版本。	推荐1366 x 768分辨率
Firefox	支持和测试最新的3个稳定版本。	
Google Chrome	支持和测试最新的3个稳定版本。	

2 设备接入

2.1 原生协议接入

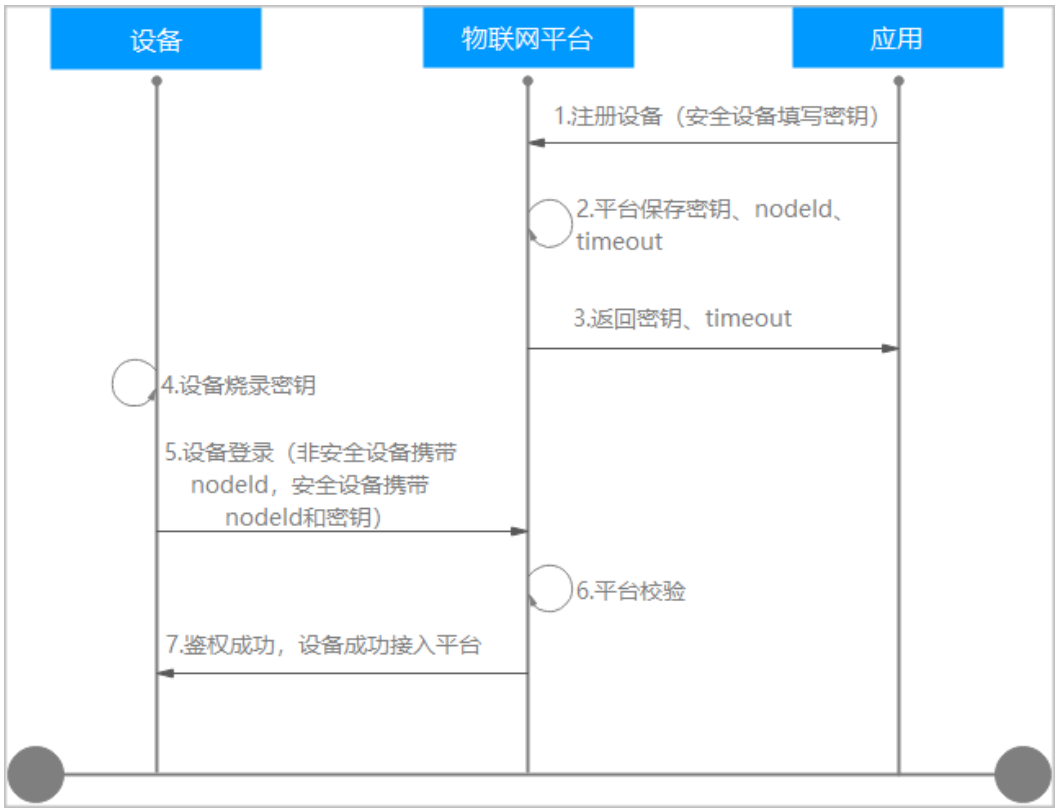
2.1.1 设备鉴权

概述

设备鉴权是指物联网平台对接入平台的设备进行身份认证。对于不同接入方式的设备，鉴权方式不同。

接入类型	鉴权方式
使用LwM2M/CoAP协议接入的设备	在设备接入物联网平台前，用户通过应用服务器调用**创建设备**接口或通过控制台在物联网平台注册设备。若为非安全设备，在设备接入物联网平台时携带设备唯一标识，完成设备的接入鉴权；当采用DTLS/DTLS+传输层安全协议接入时，即设备为安全设备时，携带密钥和nodeId完成设备的接入鉴权。**如图**所示。
使用原生MQTT/MQTTS协议接入的设备	• 使用密钥鉴权： 在设备接入物联网平台前，用户通过应用服务调用**创建设备**接口或通过控制台在物联网平台注册设备，获取设备ID和密钥，并把设备ID和密钥烧录到设备中。针对MQTTS协议的设备，需要在设备侧预置CA证书；MQTT不需要，然后在设备接入物联网平台时携带**设备ID和密钥**，完成设备的接入鉴权。**如图**所示。 • 使用证书鉴权： 在设备接入物联网平台前，用户通过控制台上传设备CA证书，然后应用服务调用**创建设备**接口或通过控制台在物联网平台注册设备，获取设备ID，并把设备ID烧录到设备中。在设备接入物联网平台时携带设备侧X.509证书，完成设备的接入鉴权。**如图**所示。

使用 LwM2M/CoAP 协议接入的鉴权流程



1. 通过调用注册接口向物联网平台发送注册请求或者在控制台上注册设备。

2~3. 物联网平台向设备分配密钥，并返回timeout。

说明

- 密钥可以在注册设备时自定义，如果没有定义，平台将自动分配预置密钥。
- timeout是指超时时间，若设备在有效时间未接入物联网平台，则平台会删除该设备的注册信息。

4. 用户将密钥烧录到设备硬件或软固件中。

5. 设备上电后，安全设备携带设备唯一标识码nodeId（如IMEI）和密钥发起接入鉴权请求；非安全设备携带设备唯一标识码发起接入鉴权请求。

6~7. 平台验证通过后，返回成功响应，设备连接物联网平台成功。

使用原生 MQTT/MQTTS 协议接入的鉴权流程

1. 通过调用注册接口向物联网平台发送注册请求或者在控制台上注册设备。

   说明

   注册时需要填写设备标识码，通常使用MAC地址，Serial No或IMEI作为nodeId。

2. 物联网平台向设备分配全局唯一的设备ID（deviceId）和密钥（secret）。

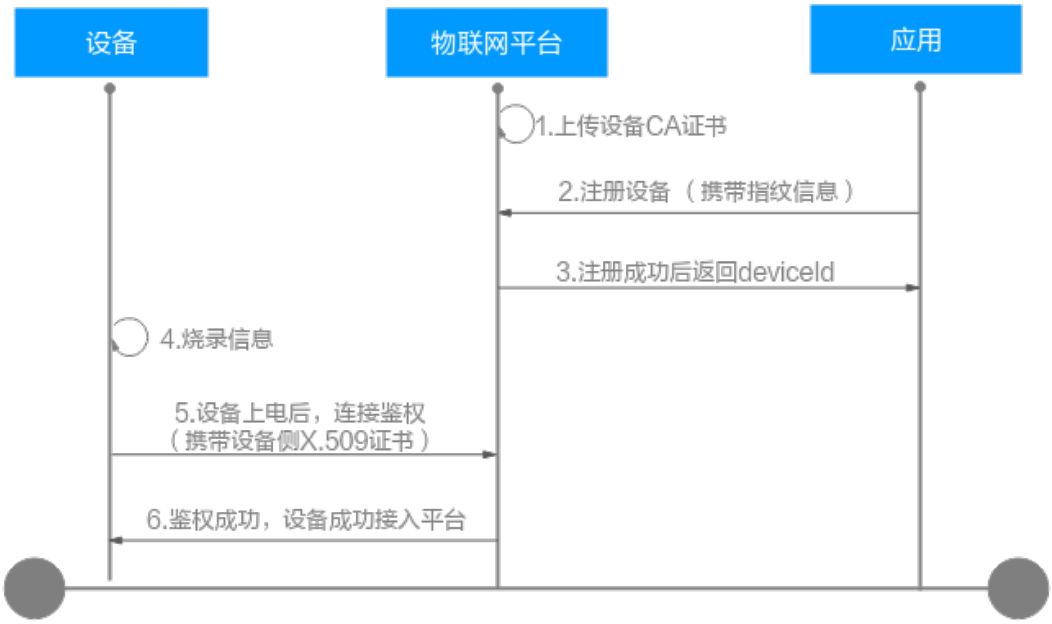
   说明

   密钥可以在注册设备时自定义，如果没有定义，平台将自动分配密钥。

3. 用户将设备ID（deviceId）和密钥（secret）烧录到设备硬件或软固件中。
4. （仅针对MQTTS协议接入的鉴权流程）设备侧需集成预置CA证书。
5. 设备上电后，携带设备ID（deviceId）和密钥（secret）发起接入鉴权请求。
6. 平台验证通过后，返回成功响应，设备连接物联网平台成功。

使用 X.509 证书的鉴权流程

X.509证书是一种用于通信实体鉴别的数字证书。当前物联网平台只支持基于MQTT协议接入的设备使用X.509证书进行设备身份认证，且每个用户最多上传100个设备CA证书。



1. 在控制台上传设备CA证书。
2. 通过调用注册接口向物联网平台发送注册请求或者在控制台上注册设备。

   说明

   注册时需要填写设备标识码，通常使用MAC地址，Serial No或IMEI作为nodeId。
3. 物联网平台向设备分配全局唯一的设备ID（deviceId）。
4. 用户将设备ID（deviceId）烧录到设备硬件或软固件中。
5. 设备上电后，携带设备侧X.509证书发起接入鉴权请求。
6. 平台验证通过后，返回成功响应，设备连接物联网平台成功。

相关 API 接口

- **创建设备**
- **重置设备密钥**
- **获取设备CA证书列表**
- **上传设备CA证书**
- **删除设备CA证书**
- **验证设备CA证书**

2.1.2 MQTT/MQTTS 协议接入

2.1.2.1 使用前必读

概述

MQTT消息由固定报头（Fixed header）、可变报头（Variable header）和有效载荷（Payload）三部分组成。

其中固定报头（Fixed header）和可变报头（Variable header）格式的填写请参考**MQTT标准规范**，有效载荷（Payload）的格式由应用定义，即设备和物联网平台之间自己定义。

说明

MQTT的语法和接口细节，请以**MQTT标准规范**为准。

常见MQTT消息类型主要有CONNECT、SUBSCRIBE、PUBLISH。

- CONNECT：指客户端请求和服务端连接。有效载荷（Payload）的主要参数，参考**设备连接鉴权**填写。
- SUBSCRIBE：指客户端订阅请求。有效载荷（Payload）中的主要参数“Topic name”，参考**Topic定义**中订阅者为设备的Topic。
- PUBLISH：平台发布消息。
  - 可变报头（Variable header）中的主要参数“Topic name”，指设备上报到物联网平台时发布者为设备的Topic。详细请参考**Topic定义**。
  - 有效载荷（Payload）中的主要参数为完整的数据上报和命令下发的消息内容，目前是一个JSON对象。

使用限制

- 上行Topic是指设备向平台发送请求，或上报数据，或回复响应。
- 下行Topic是指平台向设备下发指令，或回复响应。
- 设备与平台建立连接后，需要订阅下行Topic，否则无法收到平台下发的指令或回复的响应。应用侧接口的调用，需要设备侧的配合，例如应用侧下发命令，设备侧需要先订阅“平台命令下发”的下行Topic，否则设备无法收到平台命令，应用下发命令的接口也会报超时。

2.1.2.2 设备日志收集

2.1.2.2.1 平台下发日志收集通知

Topic

下行: $oc/devices/{device_id}/sys/events/down

接口功能

用于平台下发日志收集通知给设备。

参数说明

字段名	必选/可选	类型	参数描述
services	可选	List<EventService>	事件服务列表

EventService定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$log
event_type	必选	String	系统字段，固定为：log_config
event_time	可选	String	事件时间
paras	必选	Object	设备服务的事件列表，具体字段在设备关联的产品模型中定义。

paras参数列表

字段名	必选/可选	类型	参数描述
switch	可选	String	设备侧日志收集开关 on：开启设备侧日志收集功能 off：关闭设备侧日志收集开关
end_time	可选	String	日志收集结束时间 yyyy-MM-dd'T'HH:mm:ss'Z'

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "services": [{
        "service_id": "$log",
        "event_type": "log_config",
        "event_time": "20151212T121212Z",
        "paras": {
            "switch": "on",
            "end_time": "20151212T131212Z"
        }
    }]
}
```

2.1.2.2.2 设备上报日志内容

Topic

上行: $oc/devices/{device_id}/sys/events/up

接口功能

日志收集开关开启时设备使用该接口向平台上报日志内容，最大不超过1MB。

参数说明

字段名	必选/可选	类型	参数描述
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$log
event_type	必选	String	系统字段，固定为：log_report
event_time	可选	String	事件时间
paras	必选	Object	设备服务的事件列表，具体字段在设备关联的产品模型中定义。

paras参数列表

字段名	必选/可选	类型	参数描述
timestamp	可选	String	日志产生时间
type	必选	String	日志类型： DEVICE_STATUS ： 设备状态 DEVICE_PROPERTY ： 设备属性 DEVICE_MESSAGE ： 设备消息 DEVICE_COMMAND：设备命令
content	必选	String	日志内容。

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "services": [{
        "service_id": "$log",
        "event_type": "log_report",
        "event_time": "20151212T121212Z",
        "paras": {
            "timestamp": "1235668997",
            "type": "DEVICE_MESSAGE",
            "content": "log content"
        }
    }]
}
```

2.1.2.3 设备信息上报

2.1.2.3.1 设备信息上报

功能描述

设备向平台上报设备信息。

Topic

Topic: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$device_info
event_type	必选	String	系统字段，固定为：device_info_report
event_time	可选	String	事件时间
paras	可选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
device_sdk_version	可选	String	格式为：接入方式_版本号，例如C_v0.5.0, JAVA_v0.5.0，Tiny SDK_1.0.0等
sw_version	可选	String	软件版本
fw_version	可选	String	固件版本

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sdk_info",
        "event_type": "sdk_info_report",
        "event_time": "20151212T121212Z",
         "paras": {
             "device_sdk_version": "C_v0.5.0"
             "sw_version": "v1.0",
             "fw_version": "v1.0"
         }
    }]
}
```

2.1.2.4 设备时间同步

2.1.2.4.1 设备时间同步请求

功能描述

设备向平台发起时间同步请求。

Topic

Topic: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$time_sync
event_type	必选	String	系统字段，固定为：time_sync_request
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
device_send_time	必选	long	设备发送时间，设备获取自己的当前时间戳，即从格林威治时间1970年01月01日00时00分00秒起至现在的毫秒数

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$time_sync",
        "event_type": "time_sync_request",
        "event_time": "20151212T121212Z",
         "paras": {
            "device_send_time": 1582685678789
         }
    }]
}
```

2.1.2.4.2 设备时间同步响应

功能描述

平台向设备发送时间同步响应，携带设备发送时间device_send_time。当平台收到时间server_recv_time 后，向设备发送时间server_send_time 。

例如，设备收到的设备侧时间为device_recv_time ，则设备计算自己的准确时间为：

(server_recv_time + server_send_time + device_recv_time - device_send_time) / 2

Topic

Topic: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$time_sync
event_type	必选	String	系统字段，固定为：time_sync_response
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
device_send_time	必选	long	设备发送时间，设备获取自己的当前时间戳，即从格林威治时间1970年01月01日00时00分00秒起至现在的毫秒数
server_recv_time	必选	long	平台收到时间戳
server_send_time	必选	long	平台发送时间戳

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$time_sync",
        "event_type": "time_sync_response",
        "event_time": "20151212T121212Z",
         "paras": {
            "device_send_time": 1582685678789,
            "server_recv_time": 1582685696152,
            "server_send_time": 1582685708109
         }
    }]
}
```

2.1.2.5 文件上传/下载管理

2.1.2.5.1 设备上报获取文件上传 URL 请求

功能介绍

设备上报获取文件上传URL信息请求。

Topic

Topic: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$file_manager
event_type	必选	String	系统字段，固定为：get_upload_url
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
file_name	必选	String	待上传文件名称
file_attributes	可选	Object	文件属性，JSON格式的Object对象

示例

Topic: $oc/devices/{device_id}/sys/events/up
数据格式:

```
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$file_manager",
        "event_type": "get_upload_url",
        "event_time": "20151212T121212Z",
         "paras": {
            "file_name": "a.jpg",
            "file_attributes": {
                "hash_code": "58059181f378062f9b446e884362a526",
                "size": 1024
            }
         }
```

```
    }]
}
```

2.1.2.5.2 平台下发文件上传临时 URL

功能介绍

平台下发文件上传临时URL。

Topic

Topic: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$file_manager
event_type	必选	String	系统字段，固定为：get_upload_url_response
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
url	必选	String	文件上传URL
bucket_name	可选	String	OBS桶的名称
object_name	可选	String	OBS待上传对象名称，与file_name一致
expire	可选	Integer	URL过期时间，单位：秒

字段名	必选/可选	类型	参数描述
file_attributes	可选	Object	文件属性，JSON格式的Object对象

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$file_manager",
        "event_type": "get_upload_url_response",
        "event_time": "20151212T121212Z",
         "paras": {
            "url":"https://bucket.obs.cn-north-4.com/device_file/aGEKIpp5NAGxdP2oo90000/a.jpg?
Expires=1553162075&OSSAccessKeyId=LTAIYLScbHiV****&Signature=%2F88xdEFPukJ****%2F8****%2Bdv3io
%3D",
            "bucket_name":"bucket",
            "object_name": "c6b39067b0325db34663d3ef421a42f6_12345678_a.jpg",
            "expire":3600,
            "file_attributes": {
                "hash_code": "58059181f378062f9b446e884362a526",
                "size": 1024
            }
        }
    }]
}
```

2.1.2.5.3 设备上报文件上传结果

功能介绍

设备上报文件上传结果。

Topic

Topic: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$file_manager
event_type	必选	String	系统字段，固定为：upload_result_report
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
object_name	必选	String	OBS上传对象名称
result_code	必选	Integer	设备上传文件状态，结果码定义如下： ● 0：上传成功 ● 1：上传失败
status_code	可选	Integer	文件上传到OBS返回的状态码
status_description	可选	String	文件上传到OBS时状态的描述

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$file_manager",
        "event_type": "upload_result_report",
        "event_time": "20151212T121212Z",
         "paras": {
            "object_name": "c6b39067b0325db34663d3ef421a42f6_12345678_a.jpg",
            "result_code": 0,
            "status_code": 200,
            "status_description": "upload success"
         }
    }]
}
```

2.1.2.5.4 设备上报获取文件下载 URL 请求

功能介绍

设备上报获取文件下载URL信息请求到平台。

Topic

Topic: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$file_manager
event_type	必选	String	系统字段，固定为：get_download_url
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
fileName	必选	String	待下载文件名称
file_attributes	可选	Object	文件属性，JSON格式的Object对象

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$file_manager",
        "event_type": "get_download_url",
        "event_time": "20151212T121212Z",
         "paras": {
            "file_name": "a.jpg",
            "file_attributes": {
                "hash_code": "58059181f378062f9b446e884362a526",
                "size": 1024
            }
         }
    }]
}
```

2.1.2.5.5 平台下发文件下载临时 URL

功能介绍

平台下发文件下载临时URL给设备。

Topic

Topic: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$file_manager
event_type	必选	String	系统字段，固定为：get_download_url_response
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
url	必选	String	文件下载URL
bucket_name	可选	String	OBS桶的名称
object_name	可选	String	OBS待下载对象名称，与file_name一致
expire	可选	Integer	URL过期时间，单位：秒
file_attributes	可选	Object	文件属性，JSON格式的Object对象

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$file_manager",
        "event_type": "get_download_url_response",
        "event_time": "20151212T121212Z",
         "paras": {
            "url":"https://bucket.obs.cn-north-4.com/device_file/aGEKIpp5NAGxdP2oo90000/a.jpg?
Expires=1553162075&OSSAccessKeyId=LTAIYLScbHiV****&Signature=%2F88xdEFPukJ****%2F8****%2Bdv3io
%3D",
            "bucket_name":"bucket",
            "object_name": "c6b39067b0325db34663d3ef421a42f6_12345678_a.jpg",
            "expire":3600,
            "file_attributes": {
                "hash_code": "58059181f378062f9b446e884362a526",
                "size": 1024
            }

        }
    }]
}
```

2.1.2.5.6 设备上报文件下载结果

功能描述

设备上报文件下载结果。

Topic

Topic: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$file_manager
event_type	必选	String	系统字段，固定为：download_result_report
event_time	可选	String	事件时间

字段名	必选/可选	类型	参数描述
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
object_name	必选	String	OBS下载对象名称
result_code	必选	Integer	设备下载文件状态，结果码定义如下： ● 0：下载成功 ● 1：下载失败
status_code	可选	Integer	文件下载到OBS返回的状态码
status_description	可选	String	文件下载到OBS时状态的描述

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$file_manager",
        "event_type": "download_result_report",
        "event_time": "20151212T121212Z",
        "paras": {
            "object_name": "c6b39067b0325db34663d3ef421a42f6_12345678_a.jpg",
            "result_code": 0,
            "status_code": 200,
            "status_description": "download success"
        }
    }]
}
```

2.1.2.6 软固件升级

2.1.2.6.1 平台下发获取版本信息通知

功能介绍

平台下发获取版本信息通知。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$ota
event_type	必选	String	系统字段，固定为：version_query
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
--	--	--	--

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$ota",
        "event_type": "version_query",
        "event_time": "20151212T121212Z",
         "paras": {
         }
    }]
}
```

2.1.2.6.2 设备上报软固件版本

功能介绍

设备上报软固件版本信息。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$ota
event_type	必选	String	系统字段，固定为：version_report
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
sw_version	可选	String	软件版本
fw_version	可选	String	固件版本

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式:
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$ota",
        "event_type": "version_report",
        "event_time": "20151212T121212Z",
        "paras": {
            "sw_version": "v1.0",
            "fw_version": "v1.0"
        }
    }]
}
```

2.1.2.6.3 平台下发升级通知

功能介绍

物联网平台向设备侧下发升级通知。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$ota
event_type	必选	String	软固件保存在IoT平台，event_type的值如下： -固件升级：firmware_upgrade -软件升级：software_upgrade 软固件保存在OBS，event_type的值如下： -固件升级：firmware_upgrade_v2 -软件升级：software_upgrade_v2
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

event_type为firmware_upgrade、software_upgrade时paras参数列表

字段名	必选/可选	类型	参数描述
version	必选	String	软固件包版本号
url	必选	String	软固件包下载地址
file_size	必选	Integer	软固件包文件大小

字段名	必选/可选	类型	参数描述
access_token	可选	String	软固件包url下载地址的临时token
expires	可选	Integer	access_token的超期时间
sign	必选	String	软固件包SHA-256值

event_type为firmware_upgrade_v2、software_upgrade_v2时paras参数列表

字段名	必选/可选	类型	参数描述
version	必选	String	软固件包版本号
url	必选	String	软固件包下载地址(OBS地址)
expires	可选	Integer	url的超期时间

示例一

软固件保存在IoT平台,升级时设备会收到如下信息：
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：

```
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$ota",
        "event_type": "firmware_upgrade",
        "event_time": "20151212T121212Z",
        "paras": {
            "version": "v1.2",
            "url": "https://10.1.1.1:8943/iodm/inner/v1.3.0/firmwarefiles/ca1d954771ae61e5098c7f83",
            "file_size": 81362928,
            "access_token": "595124473f866b033dfa1f",
            "expires": 86400,
            "sign": "595124473f866b033dfa1f7e831c8c99a12f6143f392dfa996a819010842c99d"
        }
    }]
}
```

示例二

软固件保存在OBS中,升级时设备会收到如下信息：
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：

```
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$ota",
        "event_type": "firmware_upgrade_v2",
        "event_time": "20151212T121212Z",
        "paras": {
            "version": "v1.2",
            "url": "https://******.obs.cn-north-4.myhuaweicloud.com:443/test.bin?AccessKeyId=DX5G7W*********",
            "expires": 3600
        }
    }]
}
```

设备侧升级包下载指导

设备收到升级通知之后，通过HTTPS协议根据升级通知里面的URL下载升级包。基础版和标准版目前建议客户不校验证书可规避不能下载固件的问题。专享版如果需要校验证书，请提工单，后端帮忙配置域名。

请求方法

下载升级包的请求方法为：GET

请求消息头

附加请求消息头（header）字段，如指定的URI和HTTP方法所要求的字段。例如定义消息体类型的请求头“Content-Type”，请求鉴权信息等。

参数	说明
Content-Type	消息体的媒体类型，默认取值为“application/json”。
Authorization	访问物联网平台的认证信息，值为“Bearer {access_token}”，其中{access_token}为收到的升级通知中的access_token。

示例

```
GET https://10.1.1.1:8943/iodm/inner/v1.3.0/firmwarefiles/ca1d954771ae61e5098c7f83
Content-Type: application/json
Authorization: Bearer ******
```

注意

如果event_type为firmware_upgrade_v2、software_upgrade_v2，则在请求下载软固件包时不需要携带请求头。请求示例如下：

GET https://******.obs.cn-north-4.myhuaweicloud.com:443/test.bin?AccessKeyId=DX5G7W*********

2.1.2.6.4 设备上报升级状态

功能介绍

设备上报升级状态。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$ota
event_type	必选	String	系统字段，固定为：upgrade_progress_report
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
result_code	必选	Integer	设备的升级状态，结果码定义如下： ● 0：处理成功 ● 1：设备使用中 ● 2：信号质量差 ● 3：已经是最新版本 ● 4：电量不足 ● 5：剩余空间不足 ● 6：下载超时 ● 7：升级包校验失败 ● 8：升级包类型不支持 ● 9：内存不足 ● 10：安装升级包失败 ● 255： 内部异常
progress	可选	Integer	设备的升级进度，范围：0到100

字段名	必选/可选	类型	参数描述
version	必选	String	设备当前版本号（升级完成时上报的版本号要与上传软固件包时在平台设置的版本号一致）。
description	可选	String	升级状态描述信息，可以返回具体升级失败原因。

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$ota",
        "event_type": "upgrade_progress_report",
        "event_time": "20151212T121212Z",
        "paras": {
            "result_code": 0,
            "progress": 80,
            "version": "V2.0",
            "description": "upgrade processing"
        }
    }]
}
```

2.1.2.7 网关与子设备管理

2.1.2.7.1 平台通知网关子设备新增

功能介绍

平台将该网关新增的子设备列表信息通知给网关设备。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：add_sub_device_notify
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
devices	必选	List<DeviceInfo>	设备列表
version	必选	Long	子设备信息版本。网关设备可以保存此信息用于同步子设备列表时携带此参数。

DeviceInfo定义表

字段名	必选/可选	类型	参数描述
parent_device_id	必选	String	父节点设备ID
node_id	必选	String	设备标识码。
device_id	必选	String	设备ID
name	可选	String	设备名称
description	可选	String	设备描述
manufacturer_id	可选	String	厂商ID
model	可选	String	设备型号
product_id	可选	String	产品ID
fw_version	可选	String	固件版本
sw_version	可选	String	软件版本
status	可选	String	设备在线状态 ONLINE：设备在线 OFFLINE：设备离线 INACTIVE：设备未激活

字段名	必选/可选	类型	参数描述
extension_info	可选	Object	设备扩展信息。用户自定义的的扩展信息

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sub_device_manager",
        "event_type": "add_sub_device_notify",
        "event_time": "20151212T121212Z",
        "paras": {
            "devices": [{
                "parent_device_id": "c6b39067b0325db34663d3ef421a42f6_12345678",
                "node_id": "subdevice11",
                "device_id": "2bb4ddba-fb56-4566-8577-063ad2f5a6cc",
                "name": "subDevice11",
                "description": null,
                "manufacturer_id": "ofo",
                "model": "twx2",
                "product_id": "c6b39067b0325db34663d3ef421a42f6",
                "fw_version": null,
                "sw_version": null,
                "status": "ONLINE"
            }],
            "version": 1
        }
    }]
}
```

2.1.2.7.2 平台通知网关子设备删除

功能介绍

平台将该网关删除的子设备信息通知给网关设备。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：delete_sub_device_notify
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
devices	必选	List<DeviceInfo>	设备列表
version	必选	Long	子设备信息版本

DeviceInfo定义表

字段名	必选/可选	类型	参数描述
parent_device_id	必选	String	父节点设备ID
node_id	可选	String	设备标识。
device_id	必选	String	设备ID

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sub_device_manager",
        "event_type": "delete_sub_device_notify",
        "event_time": "20151212T121212Z",
        "paras": {
            "devices": [{
                "parent_device_id": "c6b39067b0325db34663d3ef421a42f6_12345678",
                "node_id": "subdevice11",
                "device_id": "2bb4ddba-fb56-4566-8577-063ad2f5a6cc"
            }],
            "version": 1
        }
    }]
}
```

2.1.2.7.3 网关同步子设备列表

功能介绍

网关设备从平台同步子设备列表。网关设备不在线时，平台无法将子设备新增和删除的信息及时通知到网关设备。网关设备离线再上线时，可以通过此接口从平台同步这段时间内新增或者删除的子设备信息。新增的子设备信息会通过**平台通知网关子设备新增**接口通知网关设备，删除的子设备信息会通过**平台通知网关子设备删除**通知网关设备。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<EventService>	事件服务列表

EventService定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：sub_device_sync_request
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
version	可选	Long	子设备信息版本。网关设备收到的最近一次的子设备新增或删除通知时的子设备版本信息。平台会根据此版本信息将此版本后新增或者删除的子设备信息通知给网关设备。

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式:
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sub_device_manager",
        "event_type": "sub_device_sync_request",
        "event_time": "20151212T121212Z",
        "paras": {"version": 1}
    }]
}
```

2.1.2.7.4 网关更新子设备状态

功能介绍

网关更新子设备状态。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<EventService>	事件服务列表

EventService定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：sub_device_update_status
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
device_statuses	必选	List<DeviceStatus>	设备状态列表，列表大小1~100

DeviceStatus定义表:

字段名	必选/可选	类型	参数描述
device_id	必选	String	子设备ID
status	必选	String	子设备状态。 ● OFFLINE：设备离线 ● ONLINE：设备上线

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "services": [{
        "service_id": "$sub_device_manager",
        "event_type": "sub_device_update_status",
        "event_time": "20151212T121212Z",
        "paras": {
            "device_statuses": [{
                    "device_id": "bf40f0c4-4022-41c6-a201-c5133122054a",
                    "status": "ONLINE"
                },
                {
                    "device_id": "4459c0f7-10bb-4718-9b07-7a82c2d508a5",
                    "status": "ONLINE"
                }
            ]
        }
    }]
}
```

2.1.2.7.5 网关上报获取平台产品信息消息

功能介绍

网关上报获取平台产品信息消息。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<EventService>	事件服务列表

EventService定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：get_products
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
product_ids	必选	List<String>	产品列表

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sub_device_manager",
        "event_type": "get_products",
        "event_time": "20151212T121212Z",
        "paras": {"product_ids": ["c3badd4a-2941-476b-804e-5ba9eafef331"]}
    }]
}
```

2.1.2.7.6 平台下发产品信息响应消息

功能介绍

平台下发产品信息响应消息。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<EventService>	事件服务列表

EventService定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：get_products_response
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
products	必选	List<ProudctInfo>	产品信息列表

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sub_device_manager",
        "event_type": "get_products_response",
        "event_time": "20151212T121212Z",
        "paras": {"products": []}
    }]
}
```

2.1.2.7.7 平台下发启动子设备扫描通知

功能介绍

平台下发子设备扫描命令。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<EventService>	事件服务列表

EventService定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_discovery
event_type	必选	String	系统字段，固定为：start_scan
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
protocol	可选	String	协议类型
channel	可选	String	通道
parent_device_id	可选	String	父节点Id
scan_settings	可选	JsonObject	扫描设置

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sub_device_discovery",
        "event_type": "start_scan",
        "event_time": "20151212T121212Z",
        "paras": {
            "protocol_type": "",
            "channel": "",
            "parent_device_id": "",
            "scan_settings": ""
        }
    }]
}
```

2.1.2.7.8 网关上报子设备列表信息

功能介绍

网关上报子设备列表信息。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	事件对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备
services	可选	List<EventService>	事件服务列表

EventService定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_discovery
event_type	必选	String	系统字段，固定为：scan_result
event_time	可选	String	事件时间
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
devices	可选	List<DeviceInfo>	设备列表

DeviceInfo定义表

字段名	必选/可选	类型	参数描述
parent_device_id	必选	String	父节点设备ID
node_id	必选	String	设备标识。
device_id	可选	String	设备ID
name	可选	String	设备名称
description	可选	String	设备描述
manufacturer_id	可选	String	厂商ID
model	可选	String	设备型号
product_id	可选	String	产品ID
fw_version	可选	String	固件版本
sw_version	可选	String	软件版本
status	可选	String	设备在线状态。 ● ONLINE：设备在线 ● OFFLINE：设备离线

示例

Topic: $oc/devices/{device_id}/sys/events/up
数据格式：

```
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$sub_device_discovery",
        "event_type": "scan_result",
        "event_time": "20151212T121212Z",
        "paras": {
            "devices": [{
                "parent_device_id": "c6b39067b0325db34663d3ef421a42f6_12345678",
                "node_id": "subdevice11",
                "device_id": "2bb4ddba-fb56-4566-8577-063ad2f5a6cc",
                "name": "subDevice11",
                "description": null,
                "manufacturer_id": "ofo",
                "model": "twx2",
                "product_id": "c6b39067b0325db34663d3ef421a42f6",
                "fw_version": null,
                "sw_version": null,
```

```
            "status": "ONLINE"
        }]
    }
  }]
}
```

2.1.2.7.9 网关新增子设备请求

功能介绍

网关主动新增其下接入的子设备，在平台上完成开户。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	对应网关设备id
services	必选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：add_sub_device_request
event_time	可选	String	事件时间
event_id	可选	String	事件请求Id，唯一标识这次事件请求。如果不携带该参数，该参数由物联网平台自动生成，生成规则为数字、字母、中划线组成的36位随机字符串。
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
devices	必选	List<DeviceInfo>	待新增的子设备信息列表，单次增加最大不超过50个设备

DeviceInfo定义表

字段名	必选/可选	类型	参数描述
parent_device_id	可选	String	父节点设备ID，默认为对应网关设备id
node_id	必选	String	设备标识。
device_id	可选	String	设备ID，用于唯一标识一个设备。如果携带该参数，平台将设备ID设置为该参数值；如果不携带该参数，设备ID由物联网平台分配获得，生成规则为"product_id" + "_" + "node_id"拼接而成。
name	可选	String	设备名称
description	可选	String	设备描述
product_id	必选	String	设备关联的产品ID，用于唯一标识一个产品模型，在控制台导入产品模型后由平台分配获得。
extension_info	可选	Object	设备扩展信息。用户可以自定义任何想要的扩展信息。字段值大小上限为1K。

示例

Topic: $oc/devices/{device_id}/sys/events/up
数据格式：

```
{
  "object_device_id": "{object_device_id}",
  "services": [
    {
      "service_id": "$sub_device_manager",
      "event_type": "add_sub_device_request",
      "event_time": "20151212T121212Z",
      "event_id": "40cc9ab1-3579-488c-95c6-c18941c99eb4",
      "paras": {
        "devices": [
          {
            "name": "subdevice11",
            "node_id": "subdevice11",
            "product_id": "c6b39067b0325db34663d3ef421a42f6",
            "description": "subdevice11"
          },
          {
            "name": "subdevice12",
            "node_id": "subdevice12",
```

```
        "product_id": "c6b39067b0325db34663d3ef421a42f6",
        "description": "subdevice12"
      }
    ]
  }
 }
]
}
```

2.1.2.7.10 网关新增子设备请求响应

功能介绍

平台将该网关新增的子设备列表信息通知给网关设备，平台收到网关新增子设备请求后会在30秒内通过此接口返回响应消息。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	对应网关设备id
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	$sub_device_manager
event_type	必选	String	add_sub_device_response
event_time	可选	String	事件时间
event_id	必选	String	事件请求Id，通过该参数关联对应的事件请求。
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
successful_devices	必选	List<DeviceInfo>	成功新增的子设备列表详情

字段名	必选/可选	类型	参数描述
failed_devices	必选	List<Reason>	新增子设备失败的原因

DeviceInfo定义表

字段名	必选/可选	类型	参数描述
parent_device_id	必选	String	父节点设备ID
node_id	必选	String	设备标识。
device_id	必选	String	设备ID
name	可选	String	设备名称
description	可选	String	设备描述
manufacturer_id	可选	String	厂商ID
model	可选	String	设备型号
product_id	可选	String	产品ID
fw_version	可选	String	固件版本
sw_version	可选	String	软件版本
status	可选	String	设备在线状态。 ● ONLINE：设备在线 ● OFFLINE：设备离线 ● INACTIVE：设备未激活
extension_info	可选	Object	设备扩展信息。用户可以自定义任何想要的扩展信息

Reason定义表

字段名	必选/可选	类型	参数描述
node_id	必选	String	对应请求中指定的设备的node_id
product_id	必选	String	对应请求中指定的设备的product_id
error_code	必选	String	新增失败错误原因码
error_msg	必选	String	新增失败原因描述

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
  "object_device_id": "{object_device_id}",
  "services": [
    {
      "service_id": "$sub_device_manager",
      "event_type": "add_sub_device_response",
      "event_time": "20151212T121212Z",
      "event_id": "40cc9ab1-3579-488c-95c6-c18941c99eb4",
      "paras": {
          "successful_devices": [
          {
            "device_id": "c6b39067b0325db34663d3ef421a42f6_subdevice11",
            "name": "subdevice11",
            "node_id": "subdevice11",
            "product_id": "c6b39067b0325db34663d3ef421a42f6",
            "description": "subdevice11",
            "manufacturer_id": "ofo",
            "model": "twx2",
            "fw_version": null,
            "sw_version": null,
            "status": "ONLINE",
            "extension_info" : null,
            "parent_device_id" : null
          }
        ],
        "failed_devices": [
          {
            "node_id": "subdevice12",
            "product_id": "c6b39067b0325db34663d3ef421a42f6",
            "error_code": "XXX",
            "error_msg": "XXXX"
          }
        ]

      }
    }
  ]
}
```

2.1.2.7.11 网关删除子设备请求

功能介绍

网关主动删除其下接入的子设备，在平台上完成销户。

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	对应网关设备id
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：delete_sub_device_request
event_time	可选	String	事件时间
event_id	可选	String	事件请求Id，唯一标识这次事件请求。如果不携带该参数，该参数由物联网平台自动生成，生成规则为数字、字母、中划线组成的36位随机字符串。
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
devices	必选	List<String>	待删除的子设备（设备id）列表，单次删除最大不超过50个设备

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
  "object_device_id": "{object_device_id}",
  "services": [
    {
      "service_id": "$sub_device_manager",
      "event_type": "delete_sub_device_request",
      "event_time": "20151212T121212Z",
      "event_id": "40cc9ab1-3579-488c-95c6-c18941c99eb4",
      "paras": {
        "devices": [
          "c6b39067b0325db34663d3ef421a42f6_subdevice11",
          "c6b39067b0325db34663d3ef421a42f6_subdevice12"
        ]
      }
    }
  ]
}
```

2.1.2.7.12 网关删除子设备请求响应

功能介绍

平台将该网关删除的子设备列表信息通知给网关设备。平台收到网关删除子设备请求后会在30秒内通过此接口返回响应消息。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	对应网关设备id
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	系统字段，固定为：$sub_device_manager
event_type	必选	String	系统字段，固定为：delete_sub_device_response
event_time	可选	String	事件时间
event_id	必选	String	事件请求Id，通过该参数关联对应的事件请求。
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
successful_devices	必选	List<String>	成功删除的子设备（设备id）列表
failed_devices	必选	List<Reason>	子设备删除失败的原因

Reason定义表

字段名	必选/可选	类型	参数描述
device_id	必选	String	对应请求中指定的设备的device _id

字段名	必选/可选	类型	参数描述
error_code	必选	String	删除失败错误原因码
error_msg	必选	String	删除失败原因描述

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
  "object_device_id": "{object_device_id}",
  "services": [
    {
      "service_id": "$sub_device_manager",
      "event_type": "delete_sub_device_response",
      "event_time": "20151212T121212Z",
      "event_id": "40cc9ab1-3579-488c-95c6-c18941c99eb4",
      "paras": {
        "successful_devices": [
          "c6b39067b0325db34663d3ef421a42f6_subdevice11"
        ],
        "failed_devices": [
          {
            "device_id": "c6b39067b0325db34663d3ef421a42f6_subdevice12",
            "error_code": "XXX",
            "error_msg": "XXXX"
          }
        ]
      }
    }
  ]
}
```

2.1.2.7.13 网关更新子设备请求

功能介绍

网关主动更新其下接入的子设备信息，在平台上完成子设备信息更新

Topic

上行: $oc/devices/{device_id}/sys/events/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	对应网关设备id
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	$sub_device_manager
event_type	必选	String	update_sub_device_request
event_time	可选	String	事件时间
event_id	可选	String	事件请求Id，唯一标识这次事件请求。如果不携带该参数，该参数由物联网平台自动生成，生成规则为数字、字母、中划线组成的36位随机字符串。
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
devices	必选	List<DeviceInfo>	待更新的子设备信息列表，单次更新最大不超过50个设备

DeviceInfo定义表

字段名	必选/可选	类型	参数描述
device_id	必选	String	设备ID
name	可选	String	设备名称
description	可选	String	设备描述
extension_info	可选	Object	设备扩展信息。用户自定义的的扩展信息

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式:
{
  "object_device_id": "{object_device_id}",
  "services": [
    {
      "service_id": "$sub_device_manager",
      "event_type": "update_sub_device_request",
      "event_time": "20151212T121212Z",
      "event_id": "40cc9ab1-3579-488c-95c6-c18941c99eb4",
      "paras": {
        "devices": [{
          "device_id": "2bb4ddba-fb56-4566-8577-063ad2f5a6cc",
          "name": "subDevice11",
          "description": "subDevice11",
```

```
        "extension_info":null
      }]
    }
  }
 ]
}
```

2.1.2.7.14 网关更新子设备请求响应

功能介绍

平台将该网关更新子设备信息通知给网关设备。

Topic

下行: $oc/devices/{device_id}/sys/events/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	对应网关设备id
services	可选	List<ServiceEvent>	事件服务列表

ServiceEvent定义表

字段名	必选/可选	类型	参数描述
service_id	必选	String	$sub_device_manager
event_type	必选	String	update_sub_device_response
event_time	可选	String	事件时间
event_id	必选	String	事件请求Id，通过该参数关联对应的事件请求。
paras	必选	Object	事件参数JSON对象

paras参数列表

字段名	必选/可选	类型	参数描述
successful_devices	必选	List<DeviceInfo>	成功更新的子设备信息列表
failed_devices	必选	List<Reason>	子设备更新失败的原因

DeviceInfo定义表

字段名	必选/可选	类型	参数描述
parent_device_id	必选	String	父节点设备ID
node_id	必选	String	设备标识。
device_id	必选	String	设备ID
name	可选	String	设备名称
description	可选	String	设备描述
manufacturer_id	可选	String	厂商ID
model	可选	String	设备型号
product_id	可选	String	产品ID
fw_version	可选	String	固件版本
sw_version	可选	String	软件版本
status	可选	String	设备在线状态。 ● ONLINE：设备在线 ● OFFLINE：设备离线 ● INACTIVE：设备未激活
extension_info	可选	Object	设备扩展信息。用户可以自定义任何想要的扩展信息

Reason定义表

字段名	必选/可选	类型	参数描述
device_id	必选	String	对应请求中指定的设备的device _id
error_code	必选	String	删除失败错误原因码
error_msg	必选	String	删除失败原因描述

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式:
{
  "object_device_id": "{object_device_id}",
  "services": [
    {
      "service_id": "$sub_device_manager",
      "event_type": "update_sub_device_response",
      "event_time": "20151212T121212Z",
```

```
    "event_id": "40cc9ab1-3579-488c-95c6-c18941c99eb4",
    "paras": {
      "successful_devices": [
      {
          "device_id": "c6b39067b0325db34663d3ef421a42f6_subdevice11",
          "name": "subdevice11",
          "node_id": "subdevice11",
          "product_id": "c6b39067b0325db34663d3ef421a42f6",
          "description": "subdevice11",
          "manufacturer_id": "ofo",
          "model": "twx2",
          "fw_version": null,
          "sw_version": null,
          "status": "ONLINE",
          "extension_info" : null,
          "parent_device_id" : null
        }
      ],
      "failed_devices": [
        {
          "device_id": "c6b39067b0325db34663d3ef421a42f6_subdevice12",
          "error_code": "XXX",
          "error_msg": "XXXX"
        }
      ]
    }
  }
 ]
}
```

2.1.2.8 设备属性

2.1.2.8.1 设备属性上报

功能介绍

用于设备按产品模型中定义的格式将属性数据上报给平台。属性上报和消息上报的区别，请查看**消息通信说明**。

说明

低配置且资源受限或者对网络流量有要求的设备，不适合直接构造JSON数据与物联网平台通信时，可将原始二进制数据透传到物联网平台。通过**开发编解码插件**实现二进制数据到平台定义JSON格式的转换。

Topic

上行：$oc/devices/{device_id}/sys/properties/report

参数说明

字段名	必选/可选	类型	参数描述
services	必选	List<ServiceProperty>	设备服务数据列表（具体结构参考下表ServiceProperty定义表）。

ServiceProperty结构定义：

字段名	必选/可选	类型	参数描述
service_id	必选	String	设备的服务ID。
properties	必选	Object	设备服务的属性列表，具体字段在设备关联的产品模型中定义，可以设置多个字段。
event_time	可选	String	设备采集数据UTC时间（格式可选：秒级别：yyyyMMdd'T'HHmmss'Z'，毫秒级别：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'），如：20161219T114920Z或者2020-08-12T12:12:12.333Z。 设备上报数据不带该参数或参数格式错误时，则数据上报时间以平台时间为准。

示例

```
Topic: $oc/devices/{device_id}/sys/properties/report
数据格式：
{
    "services": [{
            "service_id": "Temperature",
            "properties": {
                "value": 57,
                "value2": 60
            },
            "event_time": "20151212T121212Z"
        },
        {
            "service_id": "Battery",
            "properties": {
                "level": 80,
                "level2": 90
            },
            "event_time": "20151212T121212Z"
        }
    ]
}
```

2.1.2.8.2 网关批量设备属性上报

功能介绍

用于批量设备上报属性数据给平台。网关设备可以用此接口同时上报多个子设备的属性数据。

属性上报和消息上报的区别，请查看**消息通信说明**。

说明

网关批量设备属性上报，单批次最多可上报100个子设备的属性数据。如果子设备数量超过100个，建议分不同批次进行上报。

Topic

上行：$oc/devices/{device_id}/sys/gateway/sub_devices/properties/report

参数说明

字段名	必选/可选	类型	参数描述
devices	必选	List<DeviceProperty>	设备数据

DeviceService定义表

字段名	必选/可选	类型	参数描述
device_id	必选	String	设备ID
services	必选	List<ServiceProperty>	设备服务数据列表

ServiceProperty定义表：

字段名	必选/可选	类型	参数描述
service_id	必选	String	设备的服务ID。
properties	必选	Object	设备服务的属性列表，具体字段在产品模型里定义，可以设置多个字段。
event_time	可选	String	设备采集数据UTC时间（格式可选：秒级别：yyyyMMdd'T'HHmmss'Z'，毫秒级别：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'），如：20161219T114920Z或者2020-08-12T12:12:12.333Z。 设备上报数据不带该参数或参数格式错误时，则数据上报时间以平台时间为准。

示例

```
Topic:  $oc/devices/{device_id}/sys/gateway/sub_devices/properties/report
数据格式:
{
    "devices":[
        {
            "device_id":"bf40f0c4-4022-41c6-a201-c5133122054a",
            "services":[
                {
                    "service_id":"analog",
                    "properties":{
                        "PhV_phsA":"1",
                        "PhV_phsB":"2"
                    },
```

```
                "event_time":"20190606T121212Z"
            }
        ]
    },
    {
        "device_id":"42aa08ea-84c1-4025-a7b2-c1f6efe547c2",
        "services":[
            {
                "service_id":"analog",
                "properties":{
                    "PhV_phsA":"3",
                    "PhV_phsB":"5"
                },
                "event_time":"20190606T121212Z"
            },
            {
                "service_id":"parameter",
                "properties":{
                    "Load":"6",
                    "ImbA_strVal":"8"
                },
                "event_time":"20190606T121212Z"
            }
        ]
    }
  ]
}
```

2.1.2.8.3 平台设置设备属性

功能介绍

用于平台设置设备属性。设备的产品模型中定义了平台可向设备设置的属性，平台可调用此接口设置指定设备的属性数据。设备收到属性设置请求后，需要将执行结果返回给平台，如果设备没回响应平台会认为属性设置请求执行超时。

说明

低配置且资源受限或者对网络流量有要求的设备，不适合直接构造JSON数据与物联网平台通信时，可将原始二进制数据透传到物联网平台。通过**开发编解码插件**实现二进制数据到平台定义JSON格式的转换。

Topic

下行: $oc/devices/{device_id}/sys/properties/set/request_id={request_id}

上行: $oc/devices/{device_id}/sys/properties/set/response/request_id={request_id}

说明

- {request_id}用于唯一标识这次请求。设备侧收到下行请求的topic带该参数时，上行响应的topic需要将该参数值返回给平台。
- 设备侧订阅带{request_id}结尾的topic时，可以使用#，设备侧订阅平台设置设备属性的topic为：$oc/devices/{device_id}/sys/properties/set/#

下行请求参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	属性设置对应的目标设备ID
services	必选	List<ServiceProperty>	设备服务数据列表。

ServiceProperty结构定义：

字段名	必选/可选	类型	参数描述
service_id	必选	String	设备的服务ID。
properties	必选	Object	设备服务的属性列表，具体字段在产品模型里定义，可以设置多个字段。

上行响应参数说明

字段名	必选/可选	类型	参数描述
result_code	可选	Integer	命令的执行结果，0表示成功，其他表示失败。不带默认认为成功。
result_desc	可选	String	属性设置的响应描述。

下行请求示例

```
Topic: $oc/devices/{device_id}/sys/properties/set/request_id={request_id}
数据格式：
{
"object_device_id": "{object_device_id} ",
    "services": [{
            "service_id": "Temperature",
            "properties": {
                "value": 57,
                "value2": 60
            }
        },
        {
            "service_id": "Battery",
            "properties": {
                "level": 80,
                "level2": 90
            }
        }
    ]
}
```

上行响应示例

```
Topic：$oc/devices/{device_id}/sys/properties/set/response/request_id={request_id}
数据格式：
{
    "result_code": 0,
    "result_desc": "success"
}
```

2.1.2.8.4 平台查询设备属性

功能介绍

用于平台向设备查询属性信息。平台可调用此接口查询设备的属性数据。设备收到属性查询请求后，需要将设备的属性数据返回给平台，如果设备没回响应平台会认为属性查询请求执行超时。

说明

低配置且资源受限或者对网络流量有要求的设备，不适合直接构造JSON数据与物联网平台通信时，可将原始二进制数据透传到物联网平台。通过**开发编解码插件**实现二进制数据到平台定义JSON格式的转换。

Topic

下行: $oc/devices/{device_id}/sys/properties/get/request_id={request_id}

上行: $oc/devices/{device_id}/sys/properties/get/response/request_id={request_id}

说明

- {request_id}用于唯一标识这次请求。设备侧收到下行请求的topic带该参数时，上行响应的topic需要将该参数值返回给平台。
- 设备侧订阅带{request_id}结尾的topic时，可以使用#，设备侧订阅平台查询设备属性的topic为：$oc/devices/{device_id}/sys/properties/get/#

下行请求参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	属性查询对应的目标设备ID。
service_id	可选	String	设备的服务ID。

上行响应参数说明

字段名	必选/可选	类型	参数描述
services	可选	List<ServiceProperty>	设备服务数据列表。

ServiceProperty结构定义：

字段名	必选/可选	类型	参数描述
service_id	必选	String	设备的服务ID。
properties	必选	Object	设备服务的属性列表，具体字段在设备关联的产品模型里定义，可以设置多个字段。
event_time	可选	String	设备采集数据UTC时间（格式：yyyyMMdd'T'HHmmss'Z'），如：20161219T114920Z。 设备上报数据不带该参数或参数格式错误时，则数据上报时间以平台时间为准。

下行请求示例

```
Topic: $oc/devices/{device_id}/sys/properties/get/request_id={request_id}
数据格式：
{
    "object_device_id": "{object_device_id}",
    "service_id": "Temperature"
}
```

上行响应示例

```
Topic：$oc/devices/{device_id}/sys/properties/get/response/request_id={request_id}
数据格式：
{
    "services": [
        {
            "service_id": "Temperature",
            "properties": {
                "PhV_phsA": "1",
                "PhV_phsB": "2"
            },
            "event_time": "20190606T121212Z"
        }
    ]
}
```

2.1.2.8.5 设备侧获取平台的设备影子数据

功能介绍

用于设备向平台获取设备影子数据。用户可以通过应用服务器或物联网控制台配置设备影子预期数据，设备上线时订阅该topic，可以获取到平台设备影子数据，以此来同步设备属性期望值，从而完成设备属性值的修改。

简单交互逻辑介绍如下：

① 应用调用“配置设备影子预期数据”接口或在控制台配置设备影子数据；

② 设备侧（已完成关联Topic订阅）主动上报请求，获取平台的设备影子数据；

③ 平台响应设备请求，返回设备影子数据；

④ 设备侧解析属性期望值，并完成设备属性值修改。

Topic

上行请求: $oc/devices/{device_id}/sys/shadow/get/request_id={request_id}

下行响应: $oc/devices/{device_id}/sys/shadow/get/response/request_id={request_id}

说明

- {request_id}用于唯一标识这次请求。设备侧发起的消息带该参数时，需要保证设备侧该参数值的唯一性，可以用递增的数字或者UUID来实现。
- 设备侧订阅带{request_id}结尾的topic时，可以使用#，设备侧订阅获取平台设备影子数据响应的topic为：$oc/devices/{device_id}/sys/shadow/get/response/#

上行请求参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	需要获取设备影子的目标设备ID。
service_id	可选	String	需要获取设备影子的设备服务ID，不带的话查询所有服务ID的设备影子数据。

下行响应参数说明

字段名	必选/可选	类型	参数描述
object_device_id	必选	String	设备影子的目标设备ID。
shadow	可选	List<ShadowData>	服务影子数据。

ShadowData结构定义：

字段名	必选/可选	类型	参数描述
service_id	必选	String	设备的服务ID。
desired	可选	PropertiesData	设备影子desired区的属性列表。
reported	可选	PropertiesData	设备影子reported区的属性列表。
version	可选	Integer	设备影子版本信息

PropertiesData结构定义：

字段名	必选/可选	类型	参数描述
properties	必选	Object	设备服务的属性列表，具体字段在设备关联的产品模型里定义，可以设置多个字段。
event_time	可选	String	设备属性数据的UTC时间（格式：yyyyMMdd'T'HHmmss'Z'），如：20161219T114920Z。

上行请求示例

```
Topic: $oc/devices/{device_id}/sys/shadow/get/request_id={request_id}
数据格式：
{
    "object_device_id": "40fe3542-f4cc-4b6a-98c3-61a49ba1acd4",
    "service_id": "WaterMeter"
}
```

下行响应示例

```
Topic：$oc/devices/{device_id}/sys/shadow/get/response/request_id={request_id}
数据格式：
{
  "object_device_id": "40fe3542-f4cc-4b6a-98c3-61a49ba1acd4",
  "shadow": [
    {
      "service_id": "WaterMeter",
      "desired": {
        "properties": {
          "temperature": "60"
        },
        "event_time": "20151212T121212Z"
      },
      "reported": {
        "properties": {
          "temperature": "60"
        },
        "event_time": "20151212T121212Z"
      },
      "version": 1
    }
  ]
}
```

2.1.2.9 设备消息

2.1.2.9.1 设备消息上报

功能描述

是指设备无法按照产品模型中定义的属性格式进行数据上报时，可调用此接口将设备的自定义数据上报给平台，平台将设备上报的消息转发给应用服务器或华为云其他云服务上进行存储和处理。

消息上报和属性上报的区别，请查看**消息通信说明**。

说明

低配置且资源受限或者对网络流量有要求的设备，不适合直接构造JSON数据与物联网平台通信时，可将原始二进制数据透传到物联网平台。通过**开发编解码插件**实现二进制数据到平台定义JSON格式的转换。

Topic

上行：$oc/devices/{device_id}/sys/messages/up

说明

消息上报时除了使用系统预置的消息上报topic外，设备也可以使用非**平台声明**的topic。
如/usr/xxx/data/up

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	消息对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备。
name	可选	String	消息名称。
id	可选	String	消息的唯一标识。
content	必选	Object	消息内容。

说明

设备消息上报时平台不会校验消息格式，这里的参数说明和示例为给的消息样例，设备可以根据需要进行自定义数据格式的上报。

示例

```
Topic: $oc/devices/{device_id}/sys/messages/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "name": "name",
    "id": "id",
    "content": {"hello":"123"}
}
```

2.1.2.9.2 平台消息下发

功能介绍

设备无法按照产品模型中定义的格式进行指令下发时，可使用此接口下发自定义格式的数据给设备。

消息下发和命令下发的区别，请查看**消息通信说明**。

说明

低配置且资源受限或者对网络流量有要求的设备，不适合直接构造JSON数据与物联网平台通信时，可将原始二进制数据透传到物联网平台。通过**开发编解码插件**实现二进制数据到平台定义JSON格式的转换。

Topic

下行：$oc/devices/{device_id}/sys/messages/down

参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	消息对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备。
name	可选	String	消息名称
id	可选	String	消息的唯一标识
content	必选	String	消息内容。

示例

```
Topic: $oc/devices/{device_id}/sys/messages/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "name": "name",
    "id": "id",
    "content": "hello"
}
```

2.1.2.10 设备命令

2.1.2.10.1 平台命令下发

功能介绍

用于平台向设备下发设备控制命令。平台下发命令后，需要设备及时将命令的执行结果返回给平台，如果设备没回响应，平台会认为命令执行超时。命令下发和消息下发的区别，请查看**消息通信说明**。

说明

低配置且资源受限或者对网络流量有要求的设备，不适合直接构造JSON数据与物联网平台通信时，可将原始二进制数据透传到物联网平台。通过**开发编解码插件**实现二进制数据到平台定义JSON格式的转换。

Topic

下行： $oc/devices/{device_id}/sys/commands/request_id={request_id}

上行：$oc/devices/{device_id}/sys/commands/response/request_id={request_id}

说明

- {request_id}用于唯一标识这次请求。设备侧收到下行请求的topic带该参数时，上行响应的topic需要将该参数值返回给平台。
- 设备侧订阅带{request_id}结尾的topic时，可以使用通配#，设备侧订阅平台命令下发的topic为：$oc/devices/{device_id}/sys/commands/#

下行请求参数说明

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	命令对应的目标设备ID，命令下发对应的最终目标设备，没有携带则表示目标设备即topic中指定的设备。
service_id	可选	String	设备的服务ID，在设备关联的产品模型中定义
command_name	可选	String	设备命令名称，在设备关联的产品模型中定义。
paras	可选	Object	设备命令的执行参数，具体字段在设备关联的产品模型中定义。

上行响应参数说明

命令应答的json格式，具体字段在设备关联的产品模型中定义。

字段名	必选/可选	类型	参数描述
result_code	可选	Integer	标识命令的执行结果，0表示成功，其他表示失败。不带默认认为成功。
response_name	可选	String	命令的响应名称，在设备关联的产品模型中定义。
paras	可选	Object	命令的响应参数，具体字段在设备关联的产品模型中定义。

下行请求示例

```
Topic: $oc/devices/{device_id}/sys/commands/request_id={request_id}
数据格式：
{
  "object_device_id": "{object_device_id}",
  "command_name": "ON_OFF",
  "service_id": "WaterMeter",
  "paras": {
    "value": "1"
  }
}
```

上行响应示例

```
Topic：$oc/devices/{device_id}/sys/commands/response/request_id={request_id}
数据格式：
{
    "result_code": 0,
    "response_name": "COMMAND_RESPONSE",
    "paras": {
        "result": "success"
    }
}
```

2.1.2.11 设备连接鉴权

接口说明

IoT平台设备侧支持MQTT协议的connect消息接口，鉴权通过后建立设备与平台间的MQTT连接。

参数说明

参数	必选/可选	类型	参数描述
ClientId	必选	String(256)	一机一密的设备clientId由4个部分组成：设备ID、设备身份标识类型、密码签名类型、时间戳，通过下划线“_”分隔。 ● **设备ID**：指设备在平台成功注册后生成的唯一设备标识，通常由设备的产品ID和设备的NodeId通过分隔符“_”拼装而来。 ● **设备身份标识类型**：固定值为0，表示设备ID。 ● **密码签名类型**：长度1字节，当前支持2种类型： – “0”代表HMACSHA256不校验时间戳。 – “1”代表HMACSHA256校验时间戳。 ● **时间戳**：为设备连接平台时的UTC时间，格式为YYYYMMDDHH，如UTC 时间2018/7/24 17:56:20 则应表示为2018072417。
Username	必选	String(256)	设备ID。
Password	必选	String(256)	加密后的设备密钥。Password的值为使用“HMACSHA256”算法以时间戳为密钥，对secret进行加密后的值。 secret为注册设备时平台返回的secret。

设备通过MQTT协议的connect消息进行鉴权，对于构造clientId的各个部分信息都必须包括进去，平台收到connect消息时，会判断设备的鉴权类型和密码摘要算法。

- 当采用“HMACSHA256”校验时间戳方式时，会先校验消息时间戳与平台时间是否一致，再判断密码是否正确。

- 当采用“HMACSHA256”不校验时间戳方式时，鉴权消息也必须带时间戳，但不检验时间是否准确，仅判断密码是否正确。

connect消息鉴权失败时，平台会返回错误，并自动断开MQTT链路。

说明

访问**参数生成工具**，填写注册设备后生成的设备ID（DeviceId）和密钥（DeviceSecret），生成设备连接鉴权所需的参数（ClientId、Username、Password）。

原生 MQTT 协议接入建链返回码

原生MQTT协议设备和平台建链时，常见返回码如下：

返回码	返回码描述	原因
0x00	连接成功	连接成功
0x01	请求拒绝，协议版本错误	服务器不支持客户端请求MQTT协议版本
0x02	请求拒绝，无效的客户端标识符	clientId不符合格式要求或者心跳时间间隔不满足平台要求
0x03	请求拒绝，服务器不可用	平台服务不可用
0x04	请求拒绝，用户名或密码错误	用户名或密码错误
0x05	请求拒绝，没有授权	客户端没有权限连接

2.1.2.12 Topic 定义

设备使用MQTT协议接入平台时，平台和设备通过Topic进行通信。平台预置的topic列表如下：

Topic分类	Topic	Publisher(发布者)	Subscriber(订阅者)	用途
设备消息相关Topic	$oc/devices/{device_id}/sys/messages/up	设备	平台	**设备消息上报**
	$oc/devices/{device_id}/sys/messages/down	平台	设备	**平台下发消息**给设备

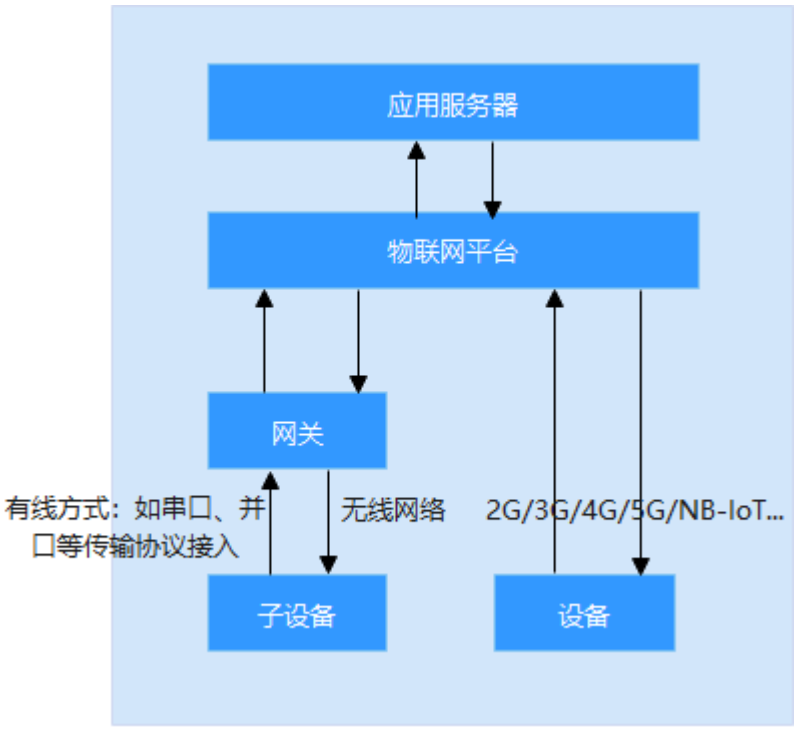
Topic分类	Topic	Publisher(发布者)	Subscriber(订阅者)	用途
设备命令相关Topic	$oc/devices/{device_id}/sys/commands/request_id={request_id}	平台	设备	**平台下发命令**给设备
	$oc/devices/{device_id}/sys/commands/response/request_id={request_id}	设备	平台	**设备返回命令响应**
设备属性相关Topic	$oc/devices/{device_id}/sys/properties/report	设备	平台	**设备上报属性**数据
	$oc/devices/{device_id}/sys/gateway/sub_devices/properties/report	设备	平台	**网关批量上报属性**数据
	$oc/devices/{device_id}/sys/properties/set/request_id={request_id}	平台	设备	**平台设置设备属性**
	$oc/devices/{device_id}/sys/properties/set/response/request_id={request_id}	设备	平台	**属性设置的响应结果**
	$oc/devices/{device_id}/sys/properties/get/request_id={request_id}	平台	设备	**平台查询设备属性**
	$oc/devices/{device_id}/sys/properties/get/response/request_id={request_id}	设备	平台	**属性查询响应结果**，这个结果不会对设备属性和影子产生影响
	$oc/devices/{device_id}/sys/shadow/get/request_id={request_id}	设备	平台	**设备侧主动获取平台的设备影子数据**
	$oc/devices/{device_id}/sys/shadow/get/response/request_id={request_id}	平台	设备	**设备侧主动获取平台设备影子数据的响应**
设备事件相关Topic	$oc/devices/{device_id}/sys/events/up	设备	平台	设备事件上报
	$oc/devices/{device_id}/sys/events/down	平台	设备	平台事件下发

说明

- {device_id}用于标识Topic路由的目标设备，设备侧订阅该topic或往topic推送消息时，该值需要替换为设备与平台建立MQTT连接时使用的设备ID参数值。
- {request_id}用于唯一标识这次请求。设备侧发起的消息带该参数时，需要保证设备侧该参数值的唯一性，可以用递增的数字或者UUID来实现。当设备收到平台下发的topic中包含request_id时，设备侧响应的request_id需要跟平台下发的保持一致。
- 设备侧订阅带{request_id}参数的topic时，可以使用通配#。如设备侧订阅命令下发请求的topic $oc/devices/{device_id}/sys/commands/request_id={request_id}时，可以用$oc/devices/{device_id}/sys/commands/#
- 平台采用了隐式订阅的功能，对于下行的系统topic，设备无需订阅，平台默认该设备订阅了qos为0的系统topic。如果需要qos为1的下行系统topic，需要设备自行订阅。

2.1.2.13 通信方式概述

设备接入物联网平台，通过物联网平台进行通信。设备、服务器和物联网平台的通信流程示意图如下。



设备发送数据到物联网平台

设备接入物联网平台后，便可与物联网平台进行通信。设备可通过以下方式发送数据到物联网平台：

- 设备消息上报：设备无法按照产品模型中定义的属性格式进行数据上报时，将设备的自定义数据通过设备消息上报接口上报给平台，平台将设备上报的消息转发给应用服务器或华为云其他云服务上进行存储和处理。
- 设备属性上报：用于设备按产品模型中定义的格式将属性数据上报给平台。

- 网关批量属性上报：用于网关设备将多个设备的属性数据一次性上报给平台。
- 设备事件上报：用于设备按产品模型中定义的格式将事件数据上报给平台。

应用服务器下发指令给设备

设备接入物联网平台后，应用服务器可通过以下方式发送指令到设备。

- 平台消息下发：用于平台下发自定义格式的数据给设备。
- 平台设置设备属性：设备的产品模型中定义了平台可向设备设置的属性，应用服务器可通过属性设置的方式修改指定设备的属性值。
- 平台查询设备属性：应用服务器通过属性查询的方式，实时查询指定设备的属性数据。
- 平台命令下发：应用服务器按产品模型中定义的命令格式下发控制命令给设备。
- 平台事件下发：应用服务器按产品模型中定义的事件格式下发事件给设备。

2.1.3 HTTPS 协议接入

2.1.3.1 使用 https 协议接入

概述

HTTPS是基于HTTP协议，通过SSL加密的一种安全通信协议。物联网平台支持HTTPS协议通信。

使用限制

描述	限制
支持的HTTP协议版本	支持 Hypertext Transfer Protocol — HTTP/1.0 协议 支持 Hypertext Transfer Protocol — HTTP/1.1 协议
支持HTTPS协议	物联网平台仅支持HTTPS协议，证书下载请参考**证书资源**。
支持的TLS版本	TLS 1.2
支持的body体最大长度	1MB
接口规格说明	请参考**产品规格说明**。
网关上报子设备属性时一次最大可上报子设备数	50

调用说明

物联网平台的Endpoint请参见：**地区和终端节点**。

说明

使用“设备接入-> HTTPS(443)”对应的Endpoint，端口为443。

HTTPS 设备与物联网平台通信

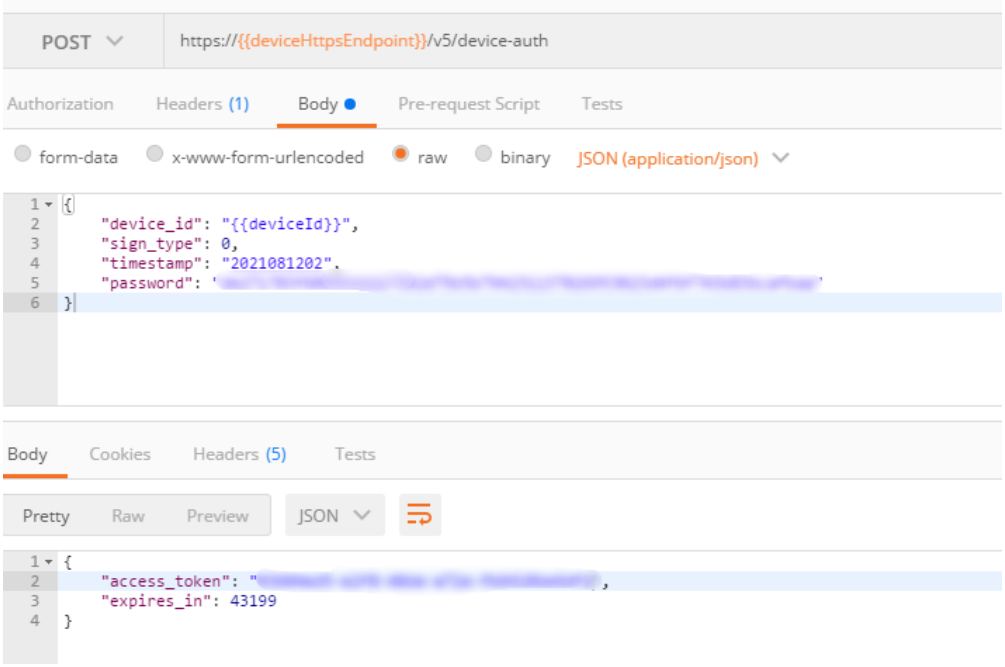
设备使用HTTPS协议接入平台时，平台和设备通过http接口调用通信。通过这些接口，平台和设备可以实现设备鉴权、消息上报及属性上报。

消息类型	说明
设备鉴权	用于设备获取鉴权信息access_token。
设备属性上报	用于设备按产品模型中定义的格式将属性数据上报给平台。
设备消息上报	设备无法按照产品模型中定义的属性格式进行数据上报时，将设备的自定义数据通过设备消息上报接口上报给平台，平台将设备上报的消息转发给应用服务器或华为云其他云服务上进行存储和处理。
网关批量属性上报	用于网关设备将多个子设备的属性数据一次性上报给平台。

业务流程

1. 设备接入前，需创建产品（可通过控制台创建或者使用应用侧API**创建产品**）。
2. 产品创建完毕后，需注册设备（可通过控制台**注册单个设备**或者使用应用侧API**注册设备**创建）。
3. 设备注册完毕后，通过设备鉴权接口获取设备的access_token。

图 2-1 获取设备 access_token



4. 获取到access_token之后，可以消息/属性上报等功能。其中access_token放于消息头中，下面示例为上报属性：

图 2-2 上报属性

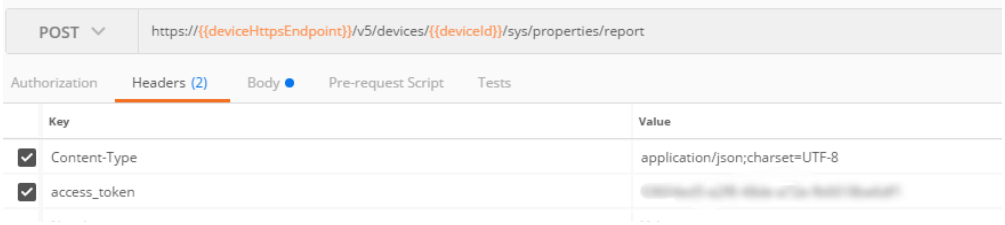
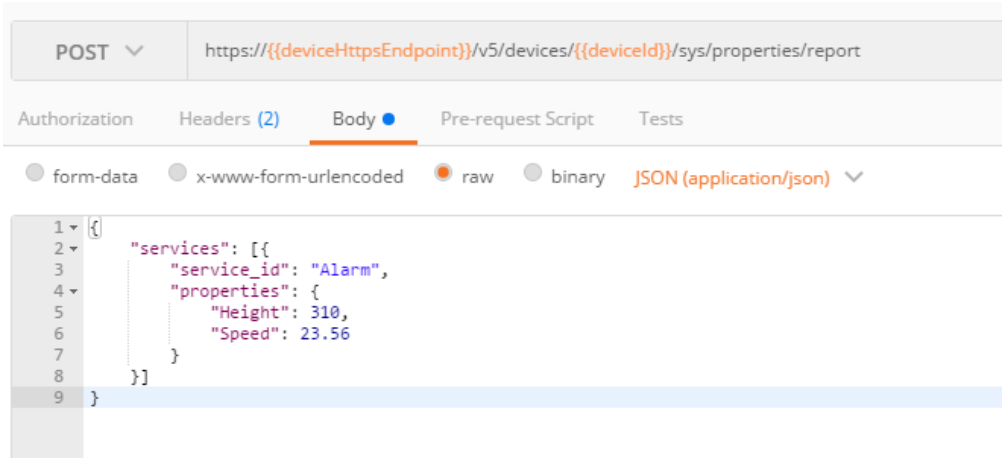


图 2-3 上报属性



http 接口介绍

物联网平台的接口如下表所示：

接口分类	用途	接口
设备鉴权相关接口	设备鉴权	/v5/device-auth
设备消息相关接口	设备消息上报	/v5/devices/{device_id}/sys/messages/up
设备属性相关接口	设备属性上报	/v5/devices/{device_id}/sys/properties/report
	网关上报子设备属性	/v5/devices/{device_id}/sys/gateway/sub-devices/properties/report

2.1.3.2 API 概览

设备鉴权

API	说明
设备鉴权	设备鉴权接口，鉴权通过后建立设备与平台间才能处理业务连接。鉴权成功后平台返回access_token。调用属性上报、消息上报等其他接口时，都需要携带access_token信息。如果access_token超期，需要重新认证设备获取access_token。如果access_token未超期重复获取access_token，老的access_token在未超期前保留30s，30s之后失效。

设备消息上报

API	说明
设备消息上报	在设备无法按照产品模型中定义的属性格式进行数据上报时，可调用此接口将设备的自定义数据上报给平台。

设备属性上报

API	说明
设备属性上报	用于设备按产品模型中定义的格式将属性数据上报给平台。
网关上报子设备属性	用于批量设备上报属性数据给平台。网关设备可以用此接口同时上报最多50个子设备的属性数据。

2.1.3.3 设备鉴权

2.1.3.3.1 设备鉴权

接口说明

设备鉴权接口，鉴权通过后建立设备与平台间才能处理业务连接。鉴权成功后平台返回access_token。调用属性上报、消息上报等其他接口时，都需要携带access_token信息。如果access_token超期，需要重新认证设备获取access_token。如果access_token未超期重复获取access_token，老的access_token在未超期前保留30s，30s之后失效。

URI

请求方法	POST
URI	/v5/device-auth

传输协议	HTTPS

请求参数

名称	必选	类型	位置	说明
device_id	是	String	Body	**参数说明: 参数说明**：设备ID，用于唯一标识一个设备。在注册设备时直接指定，或者由物联网平台分配获得。由物联网平台分配时，生成规则为"product_id" + "_" + "node_id"拼接而成。 **取值范围**：长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。 **取值范围:** 长度1-128
sign_type	是	Integer	Body	**参数说明:** 密码校验方式: 0 代表HMACSHA256校验时间戳时不会校验消息时间戳与平台时间是否一致，仅判断密码是否正确; 1 代表HMACSHA256校验时间戳时会先校验消息时间戳与平台时间是否一致，再判断密码是否正确。 **取值范围:** 大小0~1
timestamp	是	String	Body	**参数说明:** 时间戳：为设备连接平台时的UTC时间，格式为YYYYMMDDHH，如UTC 时间2018/7/24 17:56:20 则应表示为2018072417。 **取值范围:** 固定长度10
password	是	String	Body	**参数说明:** password的值为使用“HMACSHA256”算法以时间戳为秘钥，对secret进行加密后的值。secret为注册设备时平台返回的secret。 **取值范围:** 固定长度64

响应参数

名称	类型	说明
access_token	String	**参数说明:** 设备token，用于设备鉴权 **取值范围:** 长度32-256
expires_in	Integer	**参数说明:** 鉴权信息的剩余有效时间, 单位：秒

请求示例

```
POST https://{endpoint}/v5/device-auth
Content-Type: application/json

{
  "device_id" : "60a87ffebaccd902c2f1abbb_0001",
  "sign_type" : 0,
  "timestamp" : "2019120219",
  "password" : "0a5c5c4adcee661b1e730cbf6c8c343ff5924c2f3100ec2f51cad6b060183ed0"
}
```

响应示例 1

Status Code: 200 OK

```
Content-Type: application/json

{
  "access_token" : "d144a524-1997-4b99-94bf-f27128da8a34",
  "expires_in" : 86399
}
```

错误码

HTTP状态码	HTTP状态码描述	错误码	错误码描述	错误码中文描述
400	Bad Request	IOTDA.000006	Invalid input data.	请求参数不合法
401	Unauthorized	IOTDA.000002	The request is unauthorized.	鉴权失败
403	Forbidden	IOTDA.021101	Request reached the maximum rate limit.	请求已经达到限制速率
		IOTDA.021102	The request rate has reached the upper limit of the tenant, limit %s.	请求已经达到租户的限制速率

2.1.3.4 设备消息上报

2.1.3.4.1 设备消息上报

接口说明

在设备无法按照产品模型中定义的属性格式进行数据上报时，可调用此接口将设备的自定义数据上报给平台。

URI

请求方法	POST
URI	/v5/devices/{device_id}/sys/messages/up
传输协议	HTTPS

请求参数

名称	必选	类型	位置	说明
access_token	是	String	Header	**参数说明:** 调用设备鉴权信息返回的access_token **取值范围:** 长度1-256
device_id	是	String	Path	**参数说明: 参数说明**：设备ID，用于唯一标识一个设备。在注册设备时直接指定，或者由物联网平台分配获得。由物联网平台分配时，生成规则为"product_id" + "_" + "node_id"拼接而成。 **取值范围**：长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。 **取值范围:** 长度1-128
CreateDeviceMessageRequestBody	是	Object	body	需要上报的消息内容。

请求示例

```
POST https://{endpoint}/v5/devices/{device_id}/sys/messages/up
Content-Type: application/text
access_token: d144a524-1997-4b99-94bf-f27128da8a34

{
  "object_device_id" : "object_device_id",
  "name" : "name",
  "id" : "id",
  "content" : "messageUp"
}
```

响应示例 1

Status Code: 200 ok

错误码

HTTP状态码	HTTP状态码描述	错误码	错误码描述	错误码中文描述
400	Bad Request	IOTDA.000006	Invalid input data.	请求参数不合法
403	Forbidden	IOTDA.000004	Invalid access token.	非法token
		IOTDA.021101	Request reached the maximum rate limit.	请求已经达到限制速率
		IOTDA.021102	The request rate has reached the upper limit of the tenant, limit %s.	请求已经达到租户的限制速率

2.1.3.5 设备属性上报

2.1.3.5.1 设备属性上报

接口说明

用于设备按产品模型中定义的格式将属性数据上报给平台。

URI

请求方法	POST
URI	/v5/devices/{device_id}/sys/properties/report
传输协议	HTTPS

请求参数

名称	必选	类型	位置	说明
access_token	是	String	Header	**参数说明:** 调用设备鉴权信息返回的access_token **取值范围:** 长度1-256

名称	必选	类型	位置	说明
device_id	是	String	Path	**参数说明: 参数说明**：设备ID，用于唯一标识一个设备。在注册设备时直接指定，或者由物联网平台分配获得。由物联网平台分配时，生成规则为"product_id" + "_" + "node_id"拼接而成。 **取值范围**：长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。 **取值范围:** 长度1-128
services	是	List<**ServiceProperty**>	Body	**参数说明:** 设备服务数据列表

表 2-1 ServiceProperty

名称	必选	类型	说明
service_id	是	String	**参数说明:** 设备服务id
properties	是	Object	**参数说明:** 设备服务的属性列表，具体字段在设备关联的产品模型中定义
event_time	否	String	**参数说明:** 设备采集数据UTC时间（格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'），设备上报数据不带该参数或参数格式错误时，则数据上报时间以平台时间为准。

请求示例

```
POST https://{endpoint}/v5/devices/{device_id}/sys/properties/report
Content-Type: application/json
access_token: d144a524-1997-4b99-94bf-f27128da8a34

{
  "services" : [ {
    "service_id" : "serviceId",
    "properties" : {
      "Height" : 124,
      "Speed" : 23.24
    },
    "event_time" : "2021-08-13T10:10:10.555Z"
  } ]
}
```

响应示例 1

Status Code: 200 上报正常

错误码

HTTP状态码	HTTP状态码描述	错误码	错误码描述	错误码中文描述
400	Bad Request	IOTDA.000006	Invalid input data.	请求参数不合法
		IOTDA.021104	Subdevices in the request does not exist or does not belong to the gateway.	请求中有部分子设备不存在或不属于该网关.
403	Forbidden	IOTDA.000004	Invalid access token.	非法token
		IOTDA.021101	Request reached the maximum rate limit.	请求已经达到限制速率
		IOTDA.021102	The request rate has reached the upper limit of the tenant, limit %s.	请求已经达到租户的限制速率
		IOTDA.021105	The content reported in a single request cannot exceed 1 MB.	单次请求上报的内容不能超过1MB

2.1.3.5.2 网关上报子设备属性

接口说明

用于批量设备上报属性数据给平台。网关设备可以用此接口同时上报最多50个子设备的属性数据。

URI

请求方法	POST
URI	/v5/devices/{device_id}/sys/gateway/sub-devices/properties/report
传输协议	HTTPS

请求参数

名称	必选	类型	位置	说明
access_token	是	String	Header	**参数说明:** 调用设备鉴权信息返回的access_token **取值范围:** 长度1-256
device_id	是	String	Path	**参数说明: 参数说明**：设备ID，用于唯一标识一个设备。在注册设备时直接指定，或者由物联网平台分配获得。由物联网平台分配时，生成规则为"product_id" + "_" + "node_id"拼接而成。 **取值范围**：长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。 **取值范围:** 长度1-128
devices	是	List<**DeviceProperty**>	Body	**参数说明:** 设备数据列表 **取值范围:** 长度不超过50

表 2-2 DeviceProperty

名称	必选	类型	说明
device_id	是	String	**参数说明: 参数说明**：子设备的设备ID，用于唯一标识一个设备，在注册设备时由物联网平台分配获得。 **取值范围**：长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。
services	是	List<**ServiceProperty**>	**参数说明:** 设备服务数据列表

表 2-3 ServiceProperty

名称	必选	类型	说明
service_id	是	String	**参数说明:** 设备服务id
properties	是	Object	**参数说明:** 设备服务的属性列表，具体字段在设备关联的产品模型中定义
event_time	否	String	**参数说明:** 设备采集数据UTC时间（格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'），设备上报数据不带该参数或参数格式错误时，则数据上报时间以平台时间为准。

请求示例

```
POST https://{endpoint}/v5/devices/{device_id}/sys/gateway/sub-devices/properties/report
Content-Type: application/json
access_token: d144a524-1997-4b99-94bf-f27128da8a34

{
  "devices" : [ {
    "device_id" : "deviceId_0001",
    "services" : [ {
      "service_id" : "serviceId",
      "properties" : {
        "Height" : 124,
        "Speed" : 23.24
      },
      "event_time" : "2021-08-13T10:10:10.555Z"
    } ]
  }, {
    "device_id" : "deviceId_0002",
    "services" : [ {
      "service_id" : "serviceId",
      "properties" : {
        "Height" : 124,
        "Speed" : 23.24
      },
      "event_time" : "2021-08-13T10:10:10.555Z"
    } ]
  } ]
}
```

响应示例 1

Status Code: 200 上报正常

错误码

HTTP状态码	HTTP状态码描述	错误码	错误码描述	错误码中文描述
400	Bad Request	IOTDA.000006	Invalid input data.	请求参数不合法
		IOTDA.021104	Subdevices in the request does not exist or does not belong to the gateway.	请求中有部分子设备不存在或不属于该网关.
403	Forbidden	IOTDA.000004	Invalid access token.	非法token
		IOTDA.021101	Request reached the maximum rate limit.	请求已经达到限制速率

HTTP状态码	HTTP状态码描述	错误码	错误码描述	错误码中文描述
		IOTDA.021102	The request rate has reached the upper limit of the tenant, limit %s.	请求已经达到租户的限制速率
		IOTDA.021103	The request batch properties number has reached the upper limit, limit %s.	请求中子设备数量达到上限
		IOTDA.021105	The content reported in a single request cannot exceed 1 MB.	单次请求上报的内容不能超过1MB

3 设备管理

3.1 创建产品

使用物联网平台的第一步就是在控制台创建产品。产品是设备的集合，是指某一类具有相同能力或特征的设备的合集。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台”进入设备接入控制台。

步骤2 单击左侧导航栏“产品”，单击页面右上角的“创建产品”。根据页面提示填写参数，然后单击“确定”，完成产品的创建。

基本信息	
所属资源空间	下拉选择所属的资源空间。如无对应的资源空间，请先创建**资源空间**。
产品名称	为产品命名。产品名称在账号内具有唯一性。支持字母、数字、下划线（_）、连字符（-）的字符组合。
协议类型	• MQTT：使用MQTT协议接入平台的设备，数据格式可以是二进制也可以是JSON格式，采用二进制时需要部署编解码插件。 • LwM2M/CoAP：使用在资源受限（包括存储、功耗等）的NB-IoT设备，数据格式是二进制，需要部署编解码插件才能与物联网平台交互。 • Modbus：物联网平台支持使用Modbus协议接入，使用Modbus协议的设备接入IoT边缘节点的方式为非直连。直连设备和非直连设备差异说明，请参考**这里**。 • HTTP/HTTP2(TLS加密)、OPC-UA、OPC-DA、Other：通过边缘接入。
数据格式	• JSON：平台和设备之间的通信协议采用JSON格式。 • 二进制码流：您需在控制台开发**编解码插件**，将设备上报的二进制码流数据转换为JSON格式；将平台下发的JSON格式数据解析为二进制码流格式，设备才能与平台进行通信。

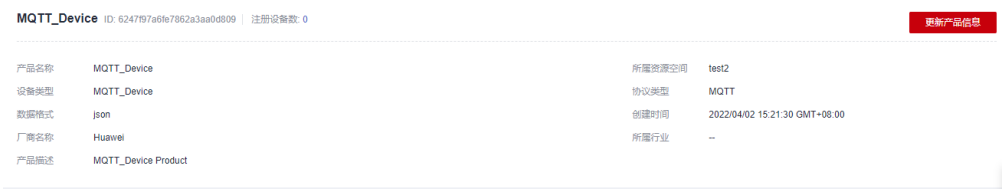
厂商名称	设备所属的厂商名称。支持中英文字符、数字、下划线（_）、连字符（-）的字符组合。
所属行业	请根据实际情况填写。
设备类型	请根据实际情况填写。
高级配置	
产品ID	定制ProductID，用于唯一标识一个产品。如果携带此参数，平台将产品ID设置为该参数值；如果不携带此参数，产品ID在物联网平台创建产品后由平台分配获得。
产品描述	产品描述。请根据实际情况填写。

产品创建成功后，您可以单击“删除”删除不再使用的产品。删除产品后，该产品下的产品模型、编解码插件等资源将被清空，请谨慎操作。

----结束

后续步骤

1. 在产品列表中，单击对应的产品，进入产品详情页。您可以查看产品ID、产品名称、设备类型、数据格式、厂商名称、所属资源空间、协议类型等产品基本信息。

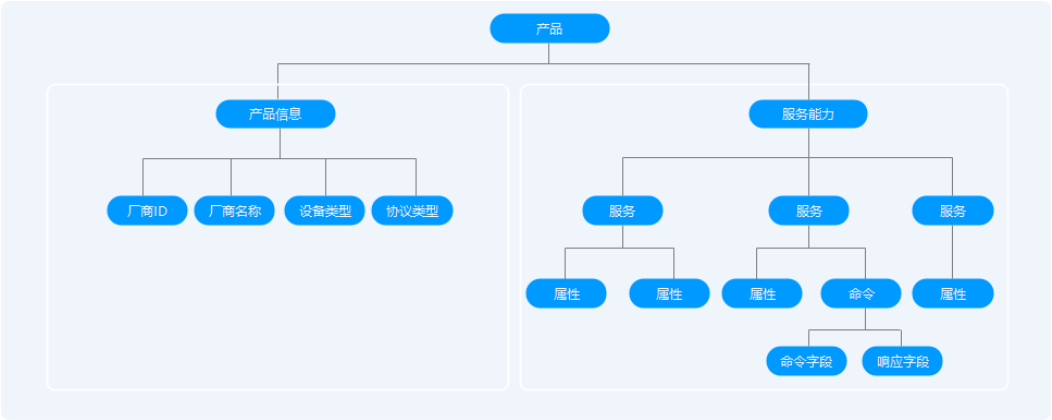


2. 您可以在产品详情页，**开发产品模型**、**开发编解码插件**、**在线调试**、**自定义Topic**。

3.2 产品模型

什么是产品模型

产品模型用于描述设备具备的能力和特性。开发者通过定义产品模型，在物联网平台构建一款设备的抽象模型，使平台理解该款设备支持的服务、属性、命令等信息，如颜色、开关等。当定义完一款产品模型后，再进行**创建设备**时，就可以使用在控制台上定义的产品模型。



产品模型包括产品信息和服务能力：

- **产品信息**

  描述一款设备的基本信息，包括厂商名称、设备类型、协议类型。

  例如：水表的厂商名称为“HZYB”，设备类型为“WaterMeter”，协议类型为“CoAP”。

- **服务能力**

  描述设备具备的业务能力。将设备业务能力拆分成若干个服务后，再定义每个服务具备的属性、命令以及命令的参数。

  以水表为例，水表具有多种能力，如上报水流、告警、电量、连接等各种数据，并且能够接受服务器下发的各种命令。产品模型文件在描述水表的能力时，可以将水表的能力划分五个服务，每个服务都需要定义各自的上报属性或命令。说明如下：

服务类型	描述
基础（WaterMeterBasic）	用于定义水表上报的水流量、水温、水压等参数，如果需要命令控制或修改这些参数，还需要定义命令的参数。
告警（WaterMeterAlarm）	用于定义水表需要上报的各种告警场景的数据，必要的话需要定义命令。
电池（Battery）	定义水表的电压、电流强度等数据。
传输规则（DeliverySchedule）	定义水表的一些传输规则，必要的话需要定义命令。
连接（Connectivity）	定义水表连接参数。

说明

具体定义几个服务是非常灵活的，如上面的例子可以将告警服务拆分成水压告警服务和流量告警服务，也可以将告警服务合入到水表基础服务中。

产品模型开发方法

物联网平台提供了多种开发产品模型的方法，您可以根据自己需求，选择对应的方法开发产品模型。

- **自定义模型（在线开发）**：从零自定义构建产品模型。详细参考**在线开发产品模型**。
- **上传模型文件（离线开发）**：将本地写好的产品模型上传到平台。详细请参考**离线开发产品模型**。
- **Excel导入**：通过导入文件的方式快速开发产品模型。详细请参考**Excel导入**。
- **导入库模型（平台预置产品模型）**：您可以使用平台预置的产品模型，快速完成产品开发。当前平台提供了标准模型和厂商模型。标准模型遵循行业标准的产品模型，适用行业内绝大部分厂商设备，而厂商模型针对设备类型发布的产品模型，适用于用行业内少量厂家设备。您可以根据实际需求选择相应的产品模型。

3.3 注册设备

3.3.1 注册单个设备

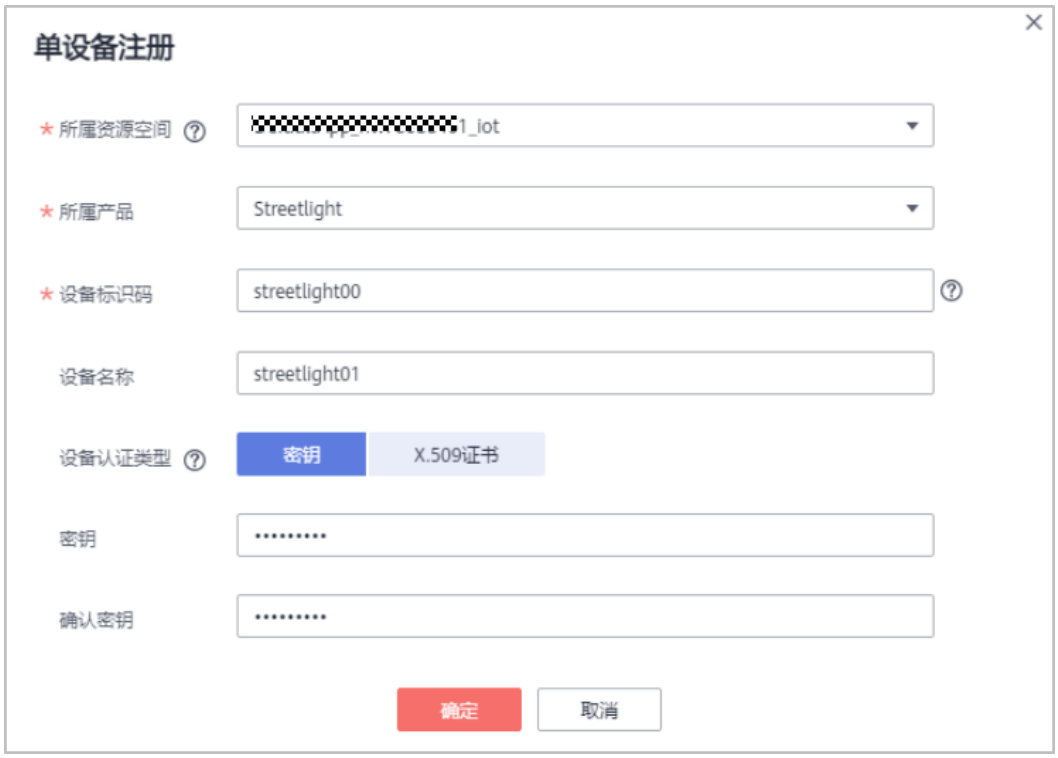
归属于某个产品下的设备实体，每个设备具有一个唯一的标识码。设备可以是直连物联网平台的设备，也可以是代理子设备连接物联网平台的网关。您可以在物联网平台注册您的实体设备，通过平台分配的设备ID和密钥，在集成了SDK后，您的设备可以接入到物联网平台，实现与平台的通信及交互。

物联网平台支持通过应用服务器调用**创建设备**接口，或者在控制台上注册单个设备。本文介绍如何在控制台上注册单个设备。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择“ 设备 > 所有设备”，单击右上角“注册设备”，按照如下表格填写参数后，单击“确定”。



参数名称	说明
所属资源空间	选择设备所属的资源空间。
所属产品	选择设备所属的产品。 只有在**这里**创建了产品，此处才可以选择具体的产品。如没有，请先创建产品。
设备标识码	即node_id，填写为设备的IMEI、MAC地址或Serial No；若没有真实设备，填写自定义字符串，由英文字母和数字组成。
设备名称	即device_name，可自定义。
设备认证类型	• 密钥：设备通过密钥验证身份。 • X.509证书：设备使用X.509证书验证身份。
密钥	设备密钥，可自定义，不填写物联网平台会自动生成。 • 若填写密钥参数，注册的是安全设备 • 若不填写密钥参数，由物联网平台生成，注册的是非安全设备
指纹	当“设备认证类型”选择“X.509证书”时填写，导入**设备侧预置的设备证书**对应的指纹，在OpenSSL执行**openssl x509 -fingerprint -sha256 -in deviceCert.pem**命令可查询。 [root@k8s-iot-wl2-2-jump cert1223]# openssl x509 -fingerprint -sha256 -in deviceCert.pem SHA256 Fingerprint=F7:91:90:45:BB:88:37:E6:A7:E7:70:4A:90:75:F3:87:DA:27:5B:7C:49:3E:FF:59:7A:6F:4F:08:4D:F8:54:E8 填写时需要删除冒号。

设备注册成功后，请妥善保管好设备ID和密钥，用于设备接入平台认证。

说明

若密钥丢失，只能通过**重置密钥**生成新的密钥，无法找回注册设备时生成的密钥。

用户可在**设备列表**删除不再使用的设备。删除设备不支持撤回，请谨慎操作。

----结束

相关 API 参考

- **查询设备列表**
- **创建设备**
- **查询设备**
- **修改设备**
- **删除设备**
- **重置设备密钥**

3.3.2 批量注册设备

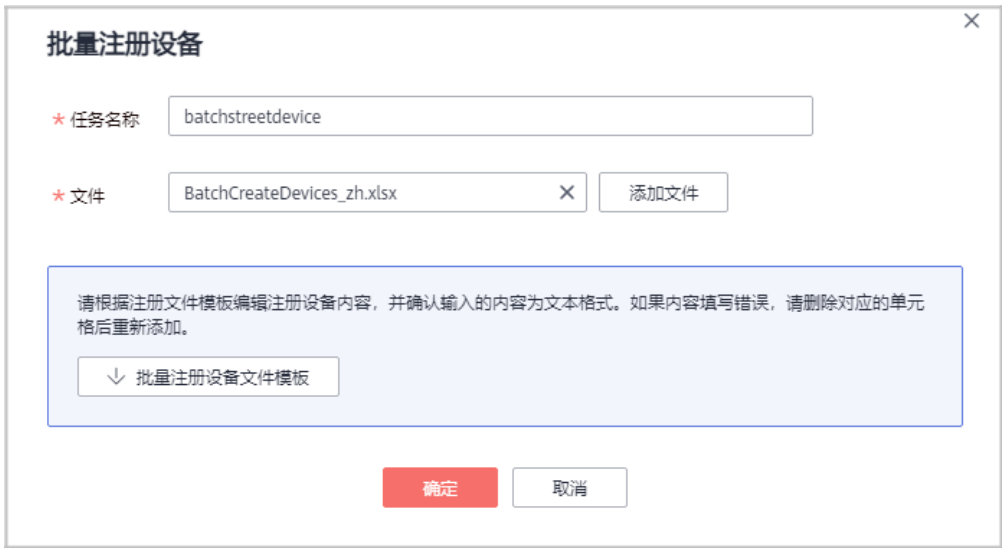
物联网平台支持通过应用服务器调用**创建批量任务**接口，或者在控制台上批量注册设备。本文介绍如何在控制台上批量注册设备。

操作步骤

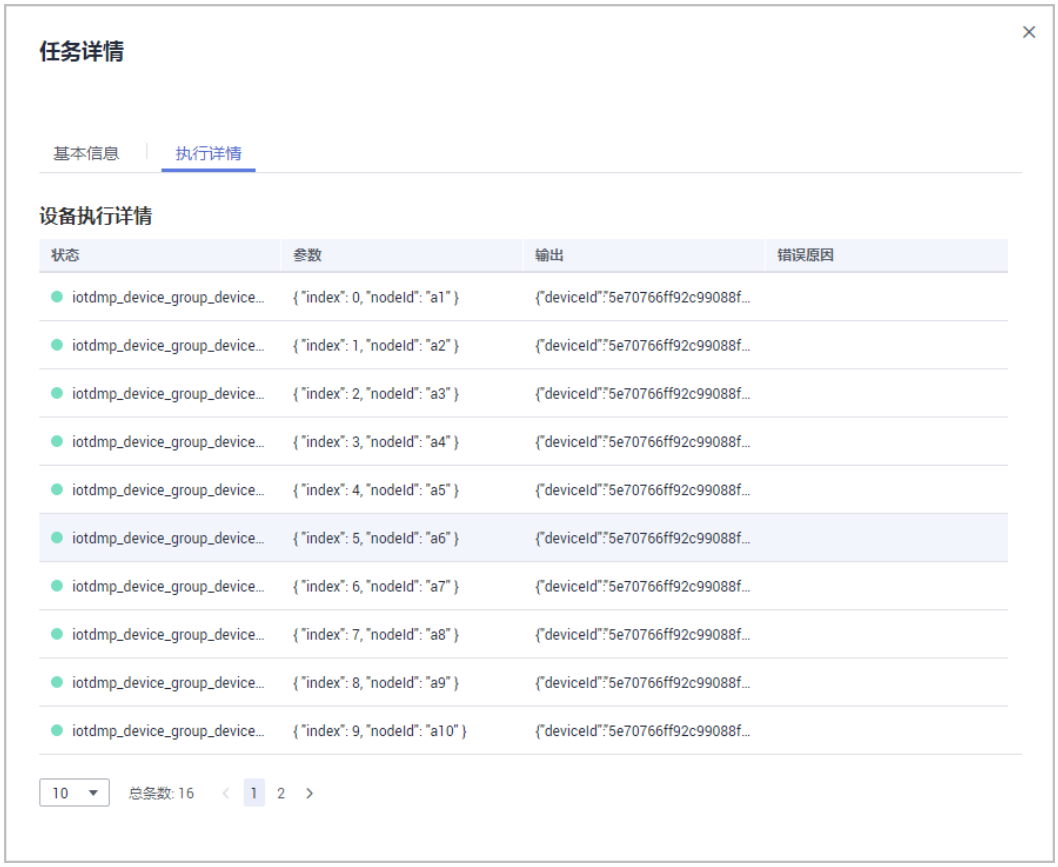
步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏，单击“ 设备 > 所有设备”。

步骤3 弹出批量注册设备窗口，先下载“批量注册设备文件模板”填写模板，然后填写“任务名称”，选择所属资源空间，并上传文件，单击“确认”完成设备的批量注册。



步骤4 批量注册执行成功，如果是原生MQTT设备注册，请单击批量任务一行，进入任务的“执行详情”，保存好设备ID和密钥，用于原生MQTT设备接入平台。



----结束

相关 API 参考

- 创建设备
- 查询批量任务列表

- 创建批量任务
- 查询批量任务

3.3.3 注册 X.509 证书认证的设备

X.509是一种用于通信实体鉴别的数字证书，物联网平台支持设备使用自己的X.509证书进行认证鉴权。使用X.509认证技术时，设备无法被仿冒，避免了密钥被泄露的风险。

注册X.509证书认证的设备前，您需要先在物联网平台上传设备的CA证书，然后在注册设备时将设备证书同设备进行绑定。本文介绍如何在物联网平台上传设备CA证书，以及注册X.509证书认证的设备。

限制说明

- 当前只有通过MQTT接入的设备支持使用X.509证书进行设备身份认证。
- 每个用户最多上传100个设备CA证书。

上传设备 CA 证书

步骤1 访问设备接入服务，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择“设备 > 设备CA证书”，单击右上角的“上传证书”。

步骤3 在弹出的对话框中，单击“添加文件”，然后单击“确认”。

说明

设备CA证书由设备厂商提供，调测时可自行制作调测证书，商用时建议更换为商用证书，否则会带来安全风险。如果已购买CA证书（pem、jks等），可直接上传到平台。

----结束

制作设备 CA 调测证书

本文以Windows环境为例，介绍通过Openssl工具制作调测证书的方法，生成的证书为PEM编码格式的证书。

1. 在浏览器中访问这里，下载并进行安装OpenSSL工具。
2. 以管理员身份运行cmd命令行窗口。
3. 执行cd c:\openssl\bin（请替换为openssl实际安装路径），进入openssl命令视图。
4. 执行以下命令生成生成密钥对。

```
openssl genrsa -out rootCA.key 2048
```

5. 执行以下命令，使用密钥对中的私有密钥生成 CA 证书。

```
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
```

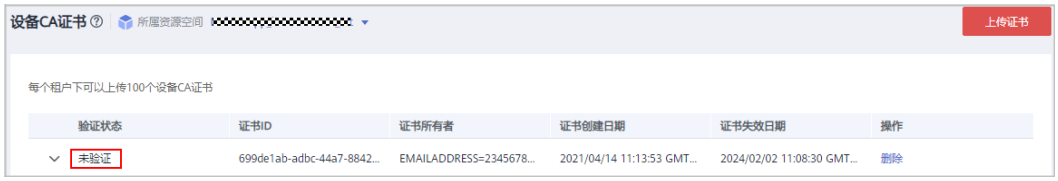
系统提示您输入如下信息，所有参数可以自定义。

- Country Name (2 letter code) [AU]：国家，如CN。
- State or Province Name (full name) []: 省份，如GD。
- Locality Name (for example, city) []:城市，如SZ。
- Organization Name (for example, company) []：组织，如Huawei。
- Organizational Unit Name (for example, section) []：组织单位，如IoT。
- Common Name (e.g. server FQDN or YOUR name) []: 名称，如zhangsan。
- Email Address []:邮箱地址，如1234567@163.com。

在openssl安装目录的bin文件夹下，获取生成的CA证书（rootCA.pem）。

上传验证证书

如果上传的是调测证书，上传后证书状态显示为“未验证”，您需要上传验证证书，来证明您拥有该CA证书。



验证证书是由设备CA证书对应的私钥创建的，请参考如下操作制作验证证书。

步骤1 执行如下命令为私有密钥验证证书生成密钥对。

```
openssl genrsa -out verificationCert.key 2048
```

步骤2 执行如下命令为私有密钥验证证书创建CSR（Certificate Signing Request）。

```
openssl req -new -key verificationCert.key -out verificationCert.csr
```

系统提示您输入如下信息，**Common Name填写为验证证书的验证码**，其他参数自定义。

- Country Name (2 letter code) [AU]：国家，如CN。
- State or Province Name (full name) []: 省份，如GD。
- Locality Name (for example, city) []:城市，如SZ。
- Organization Name (for example, company) []：组织，如Huawei。
- Organizational Unit Name (for example, section) []：组织单位，如IoT。
- Common Name (e.g. server FQDN or YOUR name) []：验证证书的验证码，请参考**步骤5**获取 。
- Email Address []:邮箱地址，如1234567@163.com。
- Password[]：密码，如1234321。
- Optional Company Name[]：公司名称，如Huawei。

步骤3 执行以下命令使用CSR创建私有密钥验证证书。

```
openssl x509 -req -in verificationCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out verificationCert.pem -days 500 -sha256
```

在openssl安装目录的bin文件夹下，获取生成的验证证书（verificationCert.pem）。

步骤4 选择对应证书，单击 ⌄ 然后单击“上传验证证书”。

步骤5 在弹出的对话框中，单击“添加文件”，然后单击“确认”。

上传验证证书后，证书状态变为“已验证”，表明您拥有该CA证书。

----结束

预置 X.509 证书

在注册X.509设备之前，您需要在设备侧预置CA机构签发的X.509证书。

说明

X.509证书由CA机构签发，若没有CA机构签发的商用证书，您可以自己**制作X.509调测证书**。如果已购买相关证书，或者权威机构签发的证书（pem、jks等），可直接上传到平台。

制作X.509调测证书

1. 以管理员身份运行cmd命令行窗口，执行cd c:\openssl\bin（请替换为openssl实际安装路径），进入openssl命令视图。
2. 执行如下命令生成密钥对。
   ```
   openssl genrsa -out deviceCert.key 2048
   ```
3. 执行如下命令为设备证书创建CSR（Certificate Signing Request）。
   ```
   openssl req -new -key deviceCert.key -out deviceCert.csr
   ```
   系统提示您输入如下信息，所有参数可以自定义。
   - Country Name (2 letter code) [AU]：国家，如CN。
   - State or Province Name (full name) []: 省份，如GD。
   - Locality Name (for example, city) []:城市，如SZ。
   - Organization Name (for example, company) []：组织，如Huawei。

- Organizational Unit Name (for example, section) []：组织单位，如IoT。
- Common Name (e.g. server FQDN or YOUR name) []: 名称，如zhangsan。
- Email Address []:邮箱地址，如1234567@163.com。
- Password[]：密码，如1234321。
- Optional Company Name[]：公司名称，如Huawei。

4. 执行以下命令使用CSR创建设备证书。

```
openssl x509 -req -in deviceCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out deviceCert.pem -days 500 -sha256
```

在openssl安装目录的bin文件夹下，获取生成的设备证书（deviceCert.pem）。

注册 X.509 证书认证的设备

步骤1 访问设备接入服务，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择“ 设备 > 所有设备”，单击右上角“注册设备”，按照如下表格填写参数后，单击“确定”。

参数名称	说明
所属资源空间	选择设备所属的资源空间。
所属产品	选择设备所属的产品。 只有在这里创建了产品，此处才可以选择具体的产品。如没有，请先创建产品。
设备标识码	即node_id，填写为设备的IMEI、MAC地址或Serial No；若没有真实设备，填写自定义字符串，由英文字母和数字组成
设备名称	即device_name，可自定义。
设备认证类型	X.509证书：设备使用X.509证书验证身份。

参数名称	说明
指纹	当“设备认证类型”选择“X.509证书”时填写，导入**设备侧预置的设备证书**对应的指纹，在OpenSSL执行**openssl x509 -fingerprint -sha256 -in deviceCert.pem**命令可查询。**注：填写时需要删除冒号。** [root@k8s-iot-wl2-2-jump cert1223]# openssl x509 -fingerprint -sha256 -in deviceCert.pem SHA256 Fingerprint=F7:91:90:45:BB:88:37:E6:A7:E7:70:4A:90:75:F3:87:DA:27:5B:7C:49:3E:FF:59:7A:6F:4F:08:4D:F8:54:E8

----结束

相关 API 参考

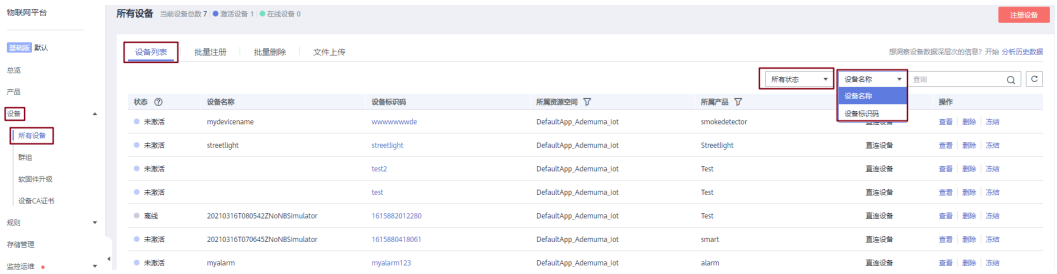
- 获取设备CA证书列表
- 上传设备CA证书
- 删除设备CA证书
- 验证设备CA证书

3.4 管理设备

在物联网平台成功创建设备后，您可以在控制台管理、查看具体设备信息、也可以冻结设备。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在设备接入控制台左侧导航栏，选择“设备 > 所有设备”，进入设备列表页，设备列表默认显示当前账号下的所有设备。



功能	描述
查看具体资源空间下设备信息	在页面上方选择您创建的资源空间。
查看具体产品下设备信息	在页面上方选择您创建的产品。
搜索设备	根据设备名称、设备标识码搜索具体设备。
查看设备信息	可查看**设备状态**、设备名称、设备标识码等，也可以单击设备对应的“查看”，查看**设备详情**。

功能	描述
删除设备	单击设备对应的“删除”。 **说明** 删除设备后，设备的相关数据也会被删除，请谨慎操作。 如果需要删除的设备数量较多，为方便节省时间，您可以调用**创建批量任务**接口，或者在控制台批量删除设备。详细请参考**批量删除设备**。
冻结设备	单击设备对应的“冻结”。 **说明** 设备冻结后不能再连接上线，当前仅支持冻结与平台直连的设备。 如果需要冻结的设备数量较多，为方便节省时间，您可以调用**创建批量任务**接口实现。
解冻设备	单击设备对应的“解冻”。 如果需要解冻的设备数量较多，为方便节省时间，您可以调用**创建批量任务**接口实现。

----结束

设备状态含义

在控制台上可以查看设备当前状态，如在线、离线、未激活、异常。用户也可以通过**订阅方式**获取设备的状态信息。设备状态定义如下：

状态	短连接设备（如NB-IoT设备）	长连接设备（MQTT）
在线	如果在25小时内设备有上报过数据，设备的状态为“在线”；如果在25小时内未上报过数据，设备的状态会变为“异常”。	设备与平台之间一直连接，无断开。
离线	设备接入平台后，设备在超过49小时未上报数据，平台会将设备置为“离线”状态。	设备与平台之间的连接断开1分钟后（数据自动刷新周期为1分钟），置为“离线”状态。 如果在界面上手动刷新状态，则直接显示“离线”。
异常	设备接入平台后，设备在超过25小时未上报数据，平台会将设备置为“异常”状态。	无此状态。
未激活	已在平台上完成设备注册但真实设备还未接入平台。请根据**设备初始化**操作完成设备的接入。	已在平台上完成设备注册但真实设备还未接入平台。请根据**设备初始化**操作完成设备的接入。

查看设备详情

在设备列表中，单击具体的设备进入到设备详情页面。

参数名	说明
概述	• 查看设备信息：查看设备基本信息，包括设备标识码（nodeId）、设备ID（deviceId），节点类型、设备软固件版本信息等。用户也可通过调用**修改设备**接口修改设备的基本信息。 – 设备标识码（nodeId），设备唯一物理标识，如IMEI、MAC地址等，用于设备在接入物联网平台时携带该标识信息完成注册鉴权。 – 设备ID（deviceId），用于唯一标识一个设备，在注册设备时由物联网平台分配获得，是设备在IoT平台上的内部标识，用于设备接入时鉴权，及后续在网络中通过deviceId进行消息传递。 • 重置密钥：密钥用于设备采用原生MQTT、NB-IoT设备、集成SDK的设备接入物联网平台的鉴权认证。重置密钥后，需要将新的密钥信息更新到设备中，设备重新发起注册时，携带新的密钥进行认证。 • 查看最新上报数据：查看最近一次设备上报到平台的数据。 • 查看历史数据：可针对此设备最近一段时间上报的历史数据进行时序洞察分析。
命令	您可以在控制台上创建单个设备的命令下发任务。详细操作参考**命令下发**。
设备影子	物联网平台提供设备影子功能，用于缓存设备状态。设备在线时，可以直接获取下发的命令；设备离线时，上线后可以主动获取下发的命令。详细操作参考**设备影子**。
消息跟踪	物联网平台支持通过消息跟踪功能进行快速的故障定位和原因分析。详细操作参考**设备消息跟踪**。
子设备	物联网平台支持设备直连，也支持设备挂载在网关上，作为网关的子设备，由网关直连。详细操作参考**网关与子设备**。
标签	物联网平台支持定义不同的标签，并对设备打标签。详细操作参考**群组与标签**。

说明

请参考**存储管理**开通物联网数据分析服务，以便查看历史数据。如果当前未开通则会自动跳转到服务开通页面。

批量删除设备

设备接入控制台也支持批量删除设备操作。操作方法如下：

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择“设备>所有设备”进入页面，单击“批量删除”页签，再单击“批量删除” 。

步骤3 在弹出批量删除设备窗口，先下载“批量删除设备文件模板”，在表格中填写需要删除的设备ID，然后填写“任务名称”，并上传文件，单击“确认”完成设备的批量删除。

界面列表显示任务执行的状态和结果。如果成功率低于100%，则可以单击具体的任务名称，进入任务详情，查看执行失败的原因。

----结束

3.5 群组与标签

概述

群组

群组是一系列设备的集合，用户可以对资源空间下所有设备，根据区域、类型等不同规则进行分类建立群组，以便处理对海量设备的批量操作。例如，对资源空间下所有水表设备的群组进行固件升级。支持群组的增删改查操作，支持给群组绑定和解绑设备，支持一个设备被添加到多个群组中。

标签管理

物联网平台支持定义不同的标签，并对设备打标签。应用服务器可调用**按标签查询资源**查询绑定了指定标签的设备。

使用限制

- 一个群组内最多添加20,000个设备。
- 一个设备最多可以被添加到10个群组中。

管理群组

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏，选择“设备 > 群组”。

步骤3 单击相关按钮，进行添加群组、解绑群组、删除群组操作。

----结束

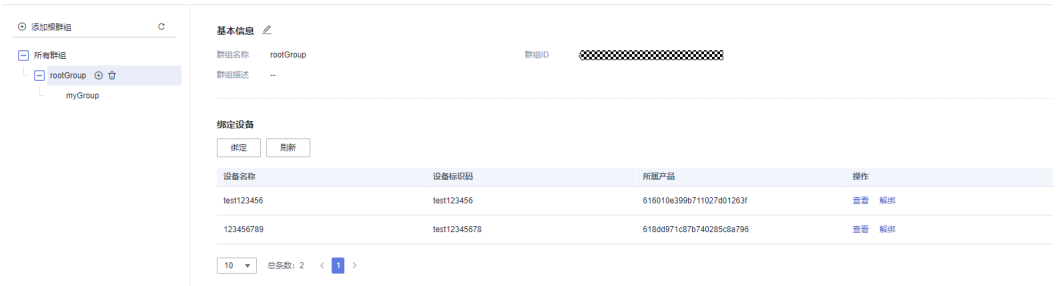
给群组绑定/解绑设备

当创建完群组后，可以进行设备和群组的绑定和解绑，也可以进行设备在不同群组之间的移动。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏，选择“设备 > 群组”。

步骤3 选择某个群组，单击"设备列表"页签，单击相关按钮，进行设备绑定、设备解绑、查看详情等操作，相关说明如下表所示。



图标	图标说明
绑定	单击可绑定设备。
解绑	选中设备后，单击“解绑”可将设备从群组中解绑。

----结束

标签

标签是一种分类方式，您可以在“设备详情”中为设备绑定标签。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择“设备”，单击“查看”进入设备详情。

步骤3 选择“标签”页签，单击“绑定标签”给设备添加标签。

----结束

相关 API 接口

绑定标签

解绑标签

按标签查询资源

查询设备组列表

添加设备组

查询设备组

修改设备组

删除设备组

管理设备组中的设备

查询设备组设备列表

3.6 设备影子

概述

物联网平台支持创建设备的“影子” 。设备影子是一个JSON文件，用于存储设备的在线状态、设备最近一次上报的设备属性值、应用服务器期望下发的配置。每个设备有且只有一个设备影子，设备可以获取和设置设备影子以此来同步设备属性值，这个同步可以是影子同步给设备，也可以是设备同步给影子。

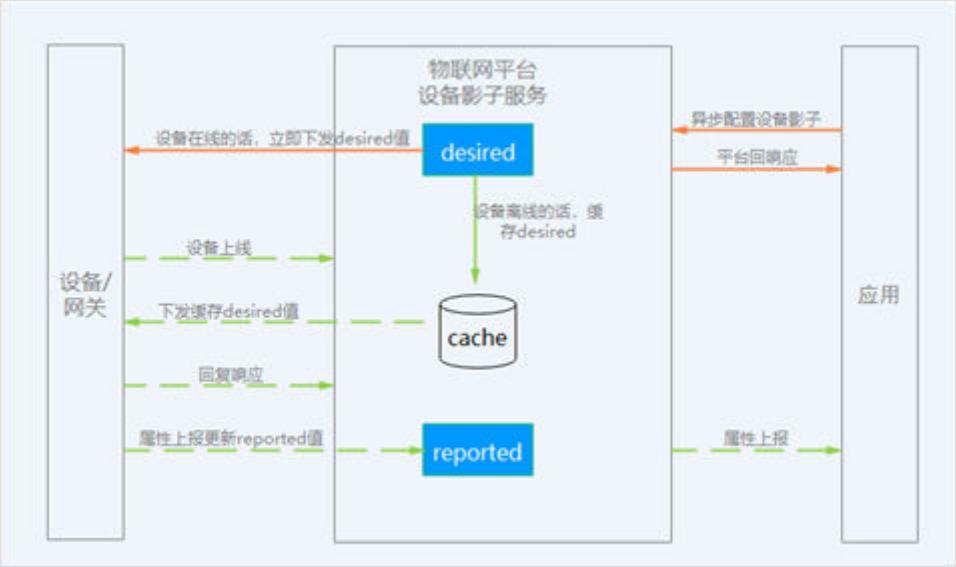
设备影子上有desired区和report区。

- desired区用于存储对设备属性的配置，即期望值。当需要修改设备的服务属性值时，可修改设备影子的desired属性值，设备在线时，desired属性值立即同步到设备。如果设备不在线，待设备上线或上报数据时，desired属性值同步到设备。

- report区用于存储设备最新上报的设备属性值，即上报值。当设备上报数据时，平台刷新report属性值为设备上报的设备属性值。

  说明

  - 设备影子可以通过调用应用侧API接口配置，也可以通过登录控制台，在设备详情->设备影子->属性配置页面配置。（设备影子主要针对设备属性配置，它的配置依赖**产品模型**）。
  - 设备影子配置属于异步命令，物联网平台会直接回复配置响应，然后平台通过设备在线状态，决定立即下发还是缓存下发。
  - 设备上线后，影子服务会下发desired值给设备，待设备属性上报时，影子服务检查属性值与下发的desired值是否匹配。若匹配，则说明影子数据在设备侧配置成功，缓存清除；若不匹配，则说明影子数据在设备侧未配置成功，在下次设备上线或属性上报时，会继续下发缓存desired值给设备，直到下发配置成功。
  - 限制：设备影子JSON文档中的key不允许特殊字符：点(.)、dollar符号($)、空char(十六进制的ASCII码为00)。如果包含了以上特殊字符则无法正常刷新影子文档。

应用场景

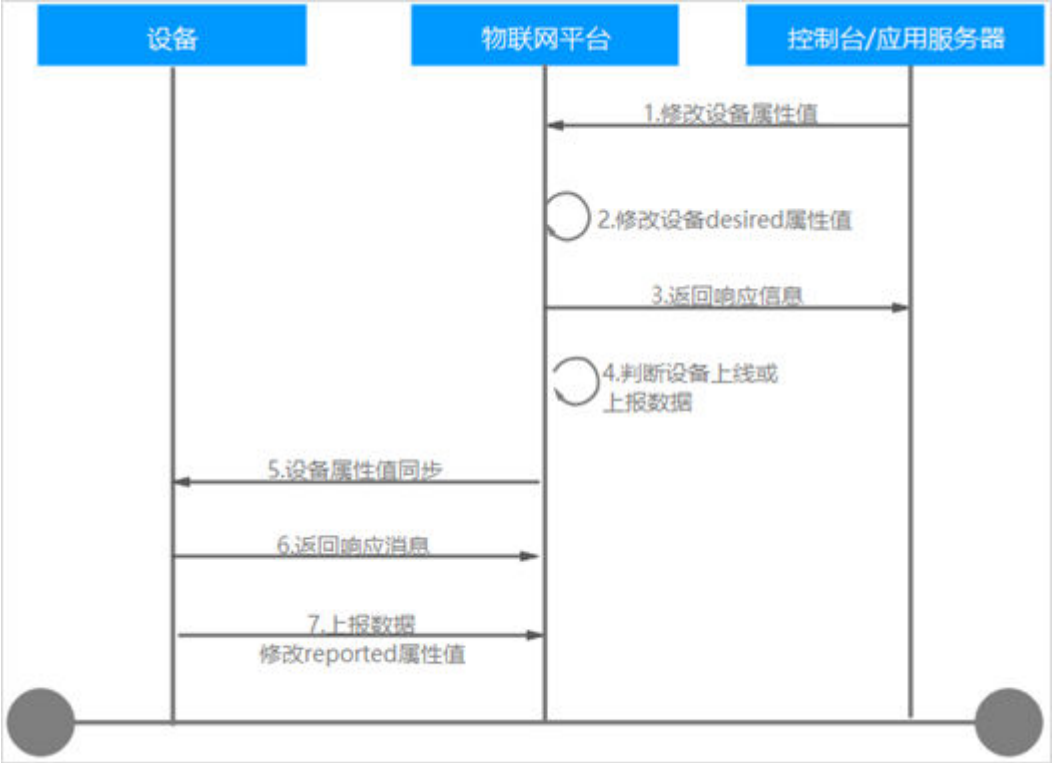
适合资源受限低功耗设备，长期处于休眠状态的场景。

- 查询设备最新上报数据和设备最新在线状态：
  - 当在控制台上查询设备上报数据时，由于设备可能长时间处于离线状态或因网络不稳定掉线，而无法获取到最新数据。通过设备影子机制，设备影子中始终保持设备最新上报的数据和设备当前状态，控制台上只需要查询设备影子中存储的数据，即可获取设备最新上报的数据和设备状态。
  - 很多应用服务器频繁的查询设备在线状态，由于设备处理能力有限，频繁查询会损耗设备性能。使用设备影子机制，设备只需要主动同步状态给设备影子一次，多个应用程序请求设备影子获取设备状态，即可获取设备最新状态，从而将应用程序和设备解耦。
- 修改设备属性值：用户通过“设备 > 设备详情 > 设备影子”修改设备的属性值。由于设备可能长时间处于离线状态，修改设备属性值的操作不能及时下发给设备。在这种情况下，物联网平台可以将修改设备的属性信息存储在设备影子中，待设备上线后，将修改的设备属性值同步给设备，从而完成设备属性值的修改。

业务流程

修改设备属性值

修改desired属性值，如果设备在线，则设备影子直接同步设备属性值到设备，否则等待设备上线或上报数据时，再同步设备属性值到设备。



1. 用户通过控制台或应用服务器修改设备属性值。消息样例如下：

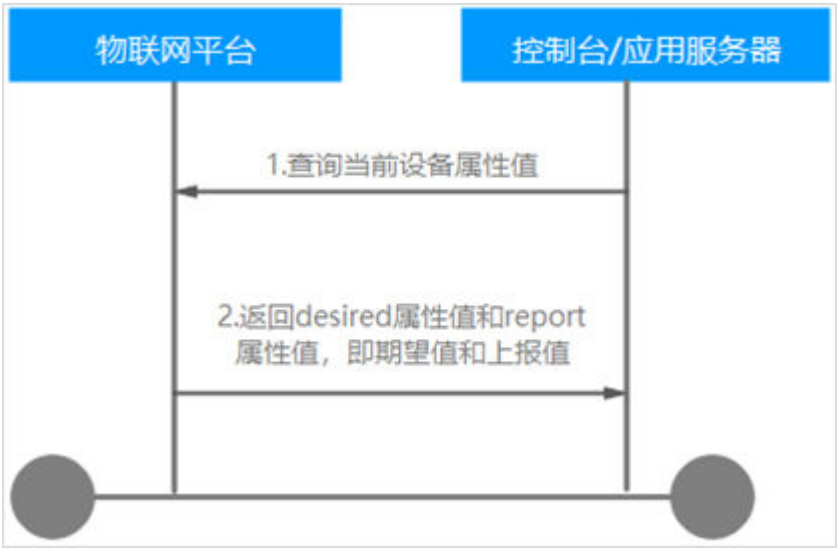
```
PUT https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}/shadow
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********

{
  "shadow" : [ {
    "desired" : {
      "temperature" : "60"
    },
    "service_id" : "WaterMeter",
    "version" : 1
  } ]
}
```

2. 物联网平台修改desired属性值。
3. 物联网平台返回响应消息。
4. 物联网平台判断设备上线或上报数据。
5. 物联网平台将设备属性同步到设备。
6. 设备返回响应消息。
7. 当设备上报数据时，物联网平台修改report属性值为设备上报的设备属性值。

查询设备属性值

设备影子保存的是设备最新的设备属性值，一旦设备属性值产生变化，设备会将设备属性值同步到设备影子。用户便可以及时获取查询结果，无需关注设备是否在线。



1. 用户通过控制台或应用服务器查询设备属性值。消息样例如下：

```
GET https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}/shadow
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

2. 物联网平台返回desired属性值和report属性值，即期望值和上报值。消息样例如下：

   Status Code: 200 OK

```
Content-Type: application/json

{
  "device_id" : "40fe3542-f4cc-4b6a-98c3-61a49ba1acd4",
  "shadow" : [ {
    "desired" : {
      "properties" : {
        "temperature" : "60"
      },
      "event_time" : "20151212T121212Z"
    },
    "service_id" : "WaterMeter",
    "reported" : {
      "properties" : {
        "temperature" : "60"
      },
      "event_time" : "20151212T121212Z"
    },
    "version" : 1
  } ]
}
```

查询和修改设备影子

查询设备影子

方法1：应用服务器调用**查询设备影子数据**接口。

方法2：登录**控制台**，在左侧导航栏选择“设备”，单击具体的设备进入到设备的详情页面，在“设备影子”页签中，可以查看当前设备属性数据，包括“上报值”和“期望值”。

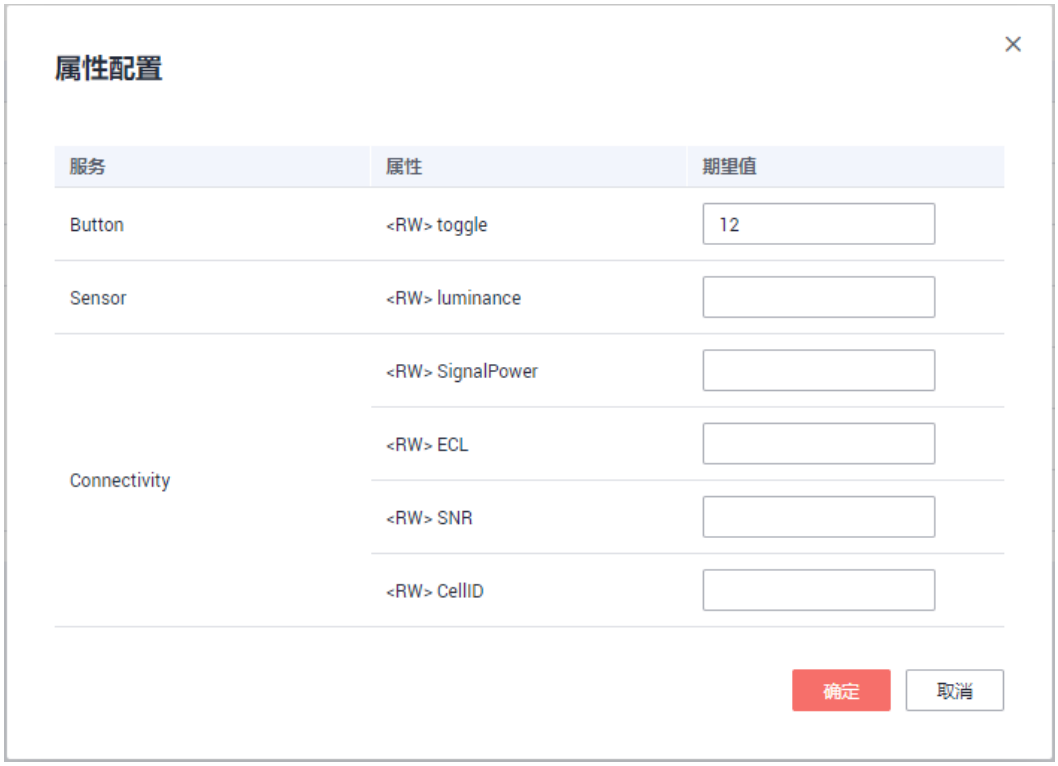
- 如果当前界面中看到“上报值”与“期望值”不一致，原因可能是设备未在线，暂时存储在设备影子中，待同步给设备，期望值会存在深色底纹。

- 如果当前界面看到的“上报值”与“期望值”一致，则表示设备最近一次上报的属性值与用户期望下发的属性值一致，期望值为白色底纹。

修改设备影子

方法1：应用服务器调用**配置设备影子预期数据**接口。

方法2：登录**控制台**，在左侧导航栏选择“设备”，单击具体的设备进入到设备的详情页面，在“设备影子”页面，单击“属性配置”，在弹出窗口中输入服务属性对应的期望值。



相关 API 接口

查询设备影子数据

配置设备影子预期数据

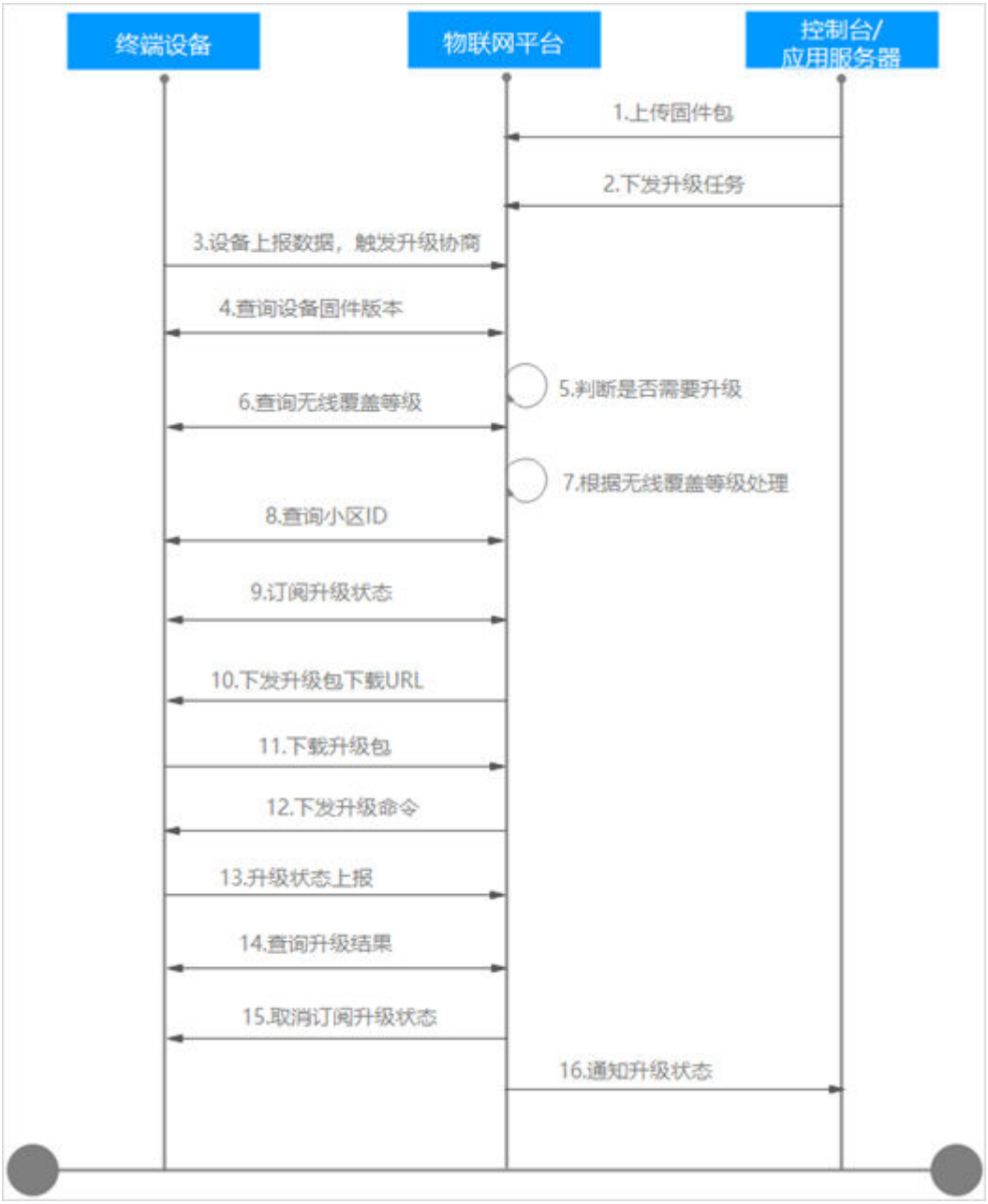
3.7 固件升级

概述

固件（Firmware）一般是指设备硬件的底层“驱动程序”，承担着一个系统最基础最底层工作的软件，比如计算机主板上的基本输入/输出系统BIOS（Basic Input/output System)。

固件升级又称为FOTA（Firmware Over The Air），是指用户可以通过OTA的方式对支持LwM2M协议和MQTT协议的设备进行固件升级。

LwM2M 协议设备固件升级流程



LwM2M协议FOTA升级流程的详细说明：

1～2. 用户在设备接入服务的控制台上传固件包，并在控制台或者应用服务器上创建固件升级任务。

3. LwM2M设备上报数据，平台感知设备上线，触发升级协商流程。(超时时间为24小时)

4～5. 物联网平台向设备下发查询设备固件版本的命令，查询成功后，物联网平台根据升级的目标版本判断设备是否需要升级。（第4步等待设备上报固件版本，超时时间为3分钟）

- 如果返回的固件版本信息与升级的目标版本信息相同，则升级流程结束，不做升级处理。
- 如果返回的固件版本信息与升级的目标版本信息不同，则继续进行下一步的升级处理。

6～7. 物联网平台查询终端设备所在的无线信号覆盖情况，获取小区ID、RSRP（Reference Signal Received Power，参考信号接收功率）和SINR（Signal to Interference Plus Noise Ratio，信号干扰噪声比）信息。（等待上报无线覆盖等级和小区ID，超时时间为3分钟左右）

- 查询成功：则根据如下方式计算可同时升级的并发数计算，并按照**步骤9**进行处理。
  - 如下图所示，如果设备的RSRP强度和SINR强度均落在等级“0”中，则同时可以对该小区的50个相同信号覆盖区间的设备进行同时升级。
  - 如果设备的RSRP强度和SINR强度分别落在等级“0”和“1”中，则以信号较弱的等级“1”为准，则只能同时对该小区的10个设备进行升级。
  - 如果设备的RSRP强度和SINR强度分别落在等级“1”和“2”中，则以信号较弱的等级“2”为准，则只能同时对该小区的1个设备进行升级。
  - 如果设备的RSRP强度和SINR强度不在该3个等级范围内，且均可以查询到，则按照信号最弱覆盖等级“2”处理，则只能同时对1个设备进行升级。

无线覆盖等级	RSRP门限范围 (dBm)	SINR门限范围 (dB)
0	-105 <= RSRP	7 <= SINR
1	-115 <= RSRP < -105	-3 <= SINR < 7
2	-125 <= RSRP < -115	-8 <= SINR < -3

说明

如果用户在固件升级中发现同时进行升级的设备数较少，则可以联系当地运营商检查和优化设备所在小区的无线覆盖情况。

- 查询失败：则按照流程**步骤8**进行处理。

8. 物联网平台继续下发查询小区ID信息的命令，获取终端设备所在的小区ID信息。

- 如果查询成功：物联网平台支持同时对该小区的10个相同情况的设备进行固件升级。
- 如果查询失败：则升级失败。

9. 物联网平台向设备订阅固件升级的状态。

10～11. 物联网平台向设备下发下载固件包的URL地址，通知设备下载固件包。终端设备根据该URL地址下载固件包，固件包的下载支持分片下载，下载完成后，设备知会物联网平台固件包已下载完毕。（第11步超时时间为60分钟）

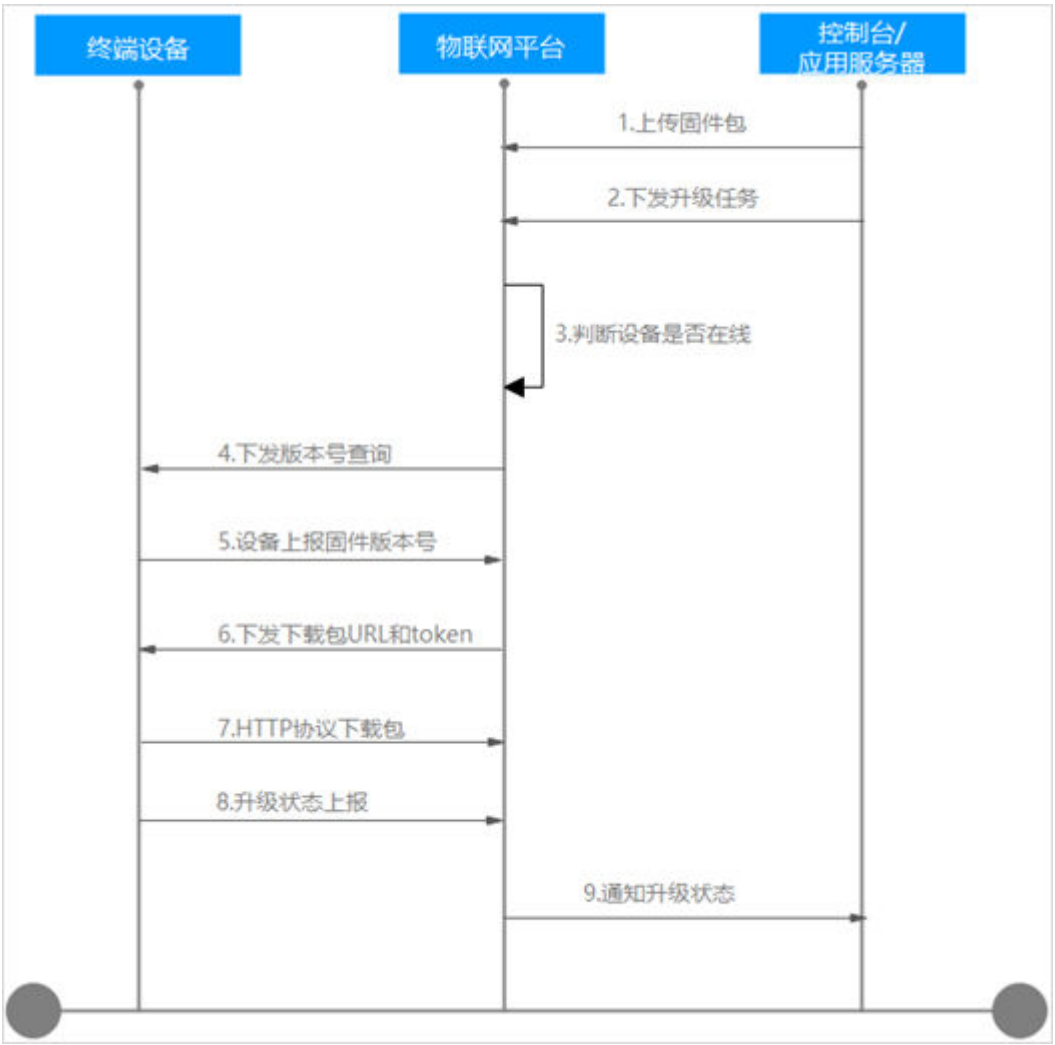
12～13. 物联网平台向设备下发升级的命令，终端设备进行升级操作，升级完成后终端设备向物联网平台反馈升级结束。（等待设备上报升级结果和升级状态，超时时间为30分钟）

14～16. 物联网平台下发命令查询固件升级的结果，获取升级结果后，向终端设备取消订阅升级状态通知，并向控制台应用服务器通知升级的结果。

说明

在下载包中断的情况下，平台支持断点续传功能。

MQTT 协议固件升级流程



MQTT协议FOTA升级流程的详细说明：

1～2. 用户在设备接入服务的控制台上传固件包，并在控制台或者应用服务器上创建固件升级任务。

3. 平台感知设备是否在线，当设备在线时立即触发升级协商流程。当设备不在线时，等待设备上线订阅升级Topic，平台感知设备上线，触发升级协商流程。（等待设备上线时间25小时以内）

4~5. 平台向设备下发查询设备固件版本号的命令，查询成功后，物联网平台根据升级的目标版本判断设备是否需要升级 。（第5步超时时间3分钟）

- 如果返回的固件版本信息与升级的目标版本信息相同，则升级流程结束，不做升级处理，升级任务置为成功。
- 如果返回的固件版本信息与升级的目标版本信息不同，且该版本号支持升级，则继续进行下一步的升级处理。

6~7. 物联网平台下发**设备侧升级包下载指导**、token及包的相关信息，用户根据下载包URL和token通过HTTPS协议来下载软件包，24小时后token无效。（下载包和升级状态上报超时时间为24小时）

8. 终端设备进行下载包升级操作，升级完成后终端设备向物联网平台反馈升级的结果。(设备升级完成后返回的版本号和设置的版本一致为成功)

9. 物联网平台向控制台/应用服务器通知升级的结果。

说明

在下载包中断的情况下，平台支持断点续传功能。

上传固件包

设备接入服务的控制台，提供固件包的管理能力，您可以将需要升级的固件包统一上传到“固件列表”界面上进行管理。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择“设备 > 软固件升级”。

步骤3 选择“固件列表”页签，单击“上传固件”。

步骤4 在弹出的页面中，按照如下信息填写参数后，单击“确认”上传固件包。

在上传固件包时，需要填写如下信息：

参数名称	说明
固件文件	添加固件包，固件包名不能带中文字符。
固件版本	固件包的版本。（设备升级完成后需要上报升级版本号，平台检查设备上报的版本号是否与此参数一致，一致为升级成功。）
选择产品	选择对应设备的**产品模型**。
支持设备源版本	手动输入，如需输入多个，可以在输入完一个版本后，单击“回车”按键，再输入下一个。

说明

目前平台仍然支持上传已签名的固件包，但无法修改公钥。如果您上传的是签过名的固件包，平台会使用之前上传的公钥对签名进行校验。如果您之前没有上传过公钥或者签名的私钥发生变化，签名会校验失败从而导致固件包上传失败。

后续平台将下线上传签名固件包功能，为了保障您能够正常使用固件升级功能，请直接上传需要下发给设备的文件。

----结束

批量设备固件升级

用户对批量设备进行固件升级有两种方式：

1. 应用服务器通过调用的“创建固件升级任务”API接口，创建批量设备的升级任务，详情请参考**创建批量任务**。
2. 通过控制台，创建批量设备的固件升级任务。

下面将重点介绍通过控制台创建批量设备的固件升级任务。

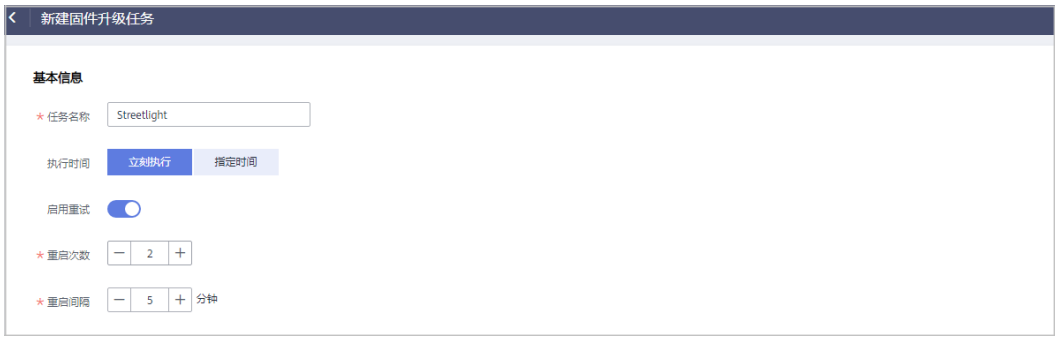
步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择"设备 > 软固件升级"，单击“升级任务”。

步骤3 在“固件升级”页签，单击“新建任务”按钮，进入新建固件任务页面。

步骤4 设置“任务信息”，填写任务名称、执行时机、启用重试。

启用重试后，可以设置重启次数和重启间隔。重启次数建议设置为2次，重启间隔设置为5分钟（最大重启次数为5次，最大重启间隔为1440分钟），即设备升级失败后，隔5分钟后会进行升级重试。



步骤5 选择需要升级的固件包。

步骤6 选择需要升级的设备群组，然后单击“提交”。

设备群组可以参考**群组与标签**创建需要升级的设备群组，并绑定对应的设备。

步骤7 创建完批量升级任务后，可以在固件升级任务列表中查看批量任务的执行结果。单击对应任务"查看"按钮，可以在“执行详情”界面查看每个设备的升级结果。

说明

如果升级任务正在执行中，是不允许删除任务的，如需删除，请先在任务列表中，手动停止任务后，再删除升级任务。

----结束

固件升级失败原因

物联网平台上报的失败原因：

失败原因	原因解释	处理建议
Device Abnormal is not online	设备异常未在线	请检查设备侧。
Task Conflict	任务冲突	请检查当前设备是否有软件升级、固件升级、日志收集或设备重启的任务正在进行。
Waiting for the device online timeout	等待设备上线超时	请检查设备侧。
Wait for the device to report upgrade result timeout	等待设备上报升级结果超时	请检查设备侧。
Waiting for report device firmware version timeout	等待上报设备固件版本超时	请检查设备侧。
Waiting for report cellId timeout	等待上报cellId超时	请检查设备侧。
Updating timeout and query device version for check timeout	等待升级结果超时，且等待设备版本信息超时	请检查设备侧。
Waiting for device downloaded package timeout	等待设备完成下载固件包超时	请检查设备侧。
Waiting for device start to update timeout	等待设备启动更新超时	请检查设备侧。
Waiting for device start download package timeout	等到设备开始下载固件包超时	请检查设备侧。

设备上报的失败原因：

失败原因	原因解释	处理建议
Not enough storage for the new firmware package	下载的固件包存储空间不足	请检查设备存储。
Out of memory during downloading process	下载过程中内存不足	请检查设备内存。
Connection lost during downloading process	下载过程中连接断开	请检查设备连接状态。
Integrity check failure for new downloaded package	下载的固件包完整性校验失败	请检查设备下载的固件包是否完整。
Unsupported package type	固件包类型不支持	请检查设备状态和厂商提供的固件包是否正确。
Invalid URI	URI不可用	检查设备侧的固件包下载地址是否正确。
Firmware update failed	固件更新失败	请检查设备侧。

常见问题

软/固件升级业务热点咨询问题如下，更多咨询问题请访问**查看更多**。

- **目标版本可以比当前版本低吗?**
- **软/固件包及其版本号如何获取?**
- **在软/固件升级任务中，业务处理是否会中断?**
- **为什么创建了软固件升级任务后，任务立即结束?**
- **常见的软/固件升级错误有哪些?**

相关 API 接口

- **创建批量任务**
- **查询批量任务列表**
- **查询批量任务**

3.8 软件升级

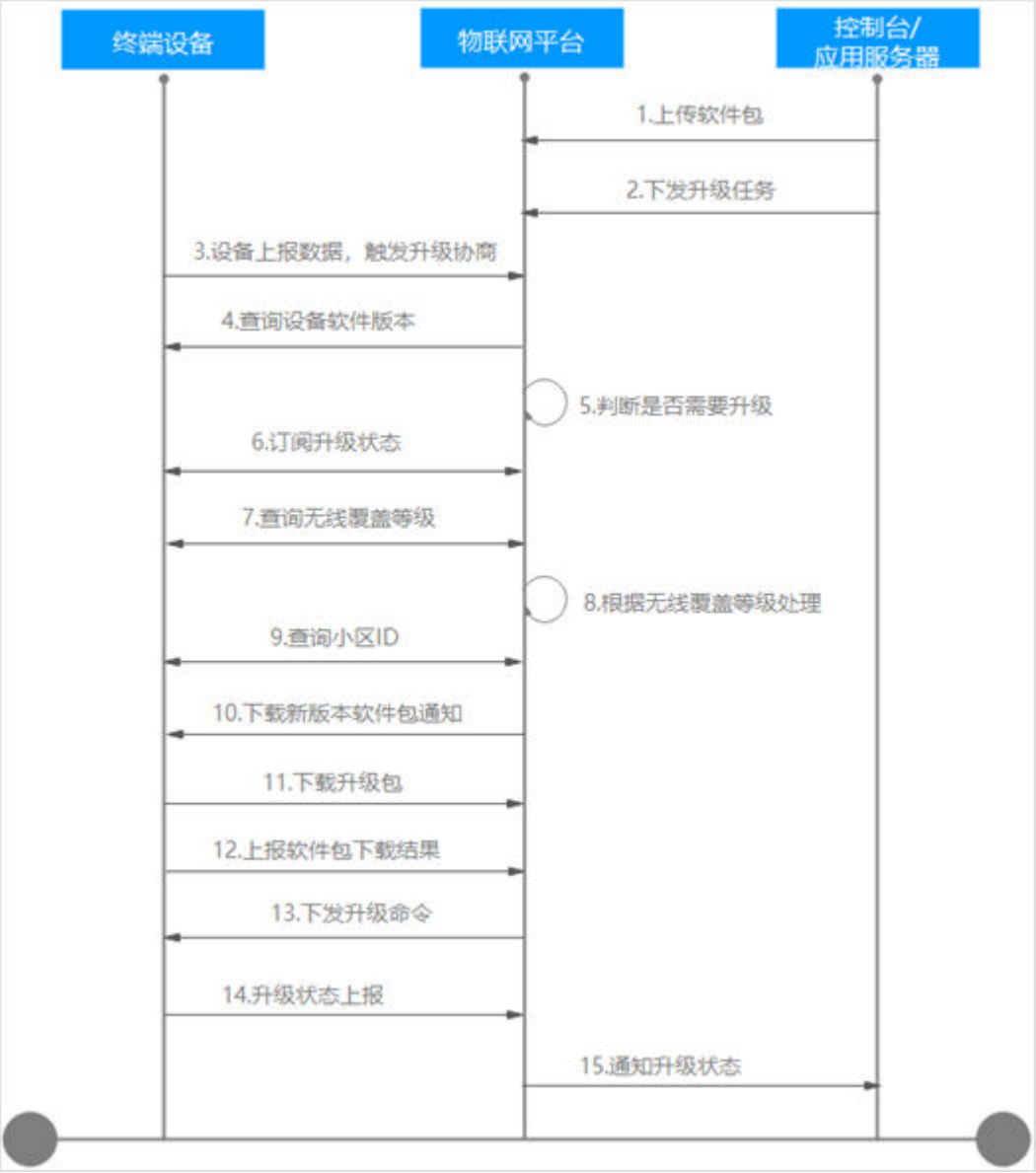
3.8.1 升级指导

概述

软件（Software）一般分为系统软件和应用软件，系统软件实现设备最基本的功能，比如编译工具、系统文件管理等；应用软件可以根据设备的特点，提供不同的功能，比如采集数据、数据分析处理等。

软件升级又称为SOTA（SoftWare Over The Air），是指用户可以通过OTA的方式支持对LwM2M协议和MQTT协议的设备进行软件升级。基于LwM2M协议的产品模型，软件升级遵循的协议为PCP协议（**查看PCP协议介绍**），设备侧需要遵循PCP协议进行软件升级的适配开发，适配方法请参考**设备侧适配开发指导**；基于MQTT协议的产品模型，不校验软件升级协议类型。

LwM2M 协议设备软件升级流程



LwM2M协议SOTA升级流程的详细说明：

1～2. 用户在设备管理服务的控制台上传软件包，并在控制台或者应用服务器上创建软件升级任务。

3. LwM2M设备上报数据，平台感知设备上线，触发升级协商流程。(超时时间为24小时)

4～5. 物联网平台向设备下发查询设备软件版本的命令，查询成功后，物联网平台根据升级的目标版本判断设备是否需要升级。（第4步等待设备上报软件版本，超时时间为3分钟）

- 如果返回的软件版本信息与升级的目标版本信息相同，则升级流程结束，不做升级处理。
- 如果返回的软件版本信息与升级的目标版本信息不同，则继续进行下一步的升级处理。

6. 物联网平台向设备订阅软件升级的状态。

7～8. 物联网平台查询终端设备所在的无线信号覆盖情况，获取小区ID、RSRP（Reference Signal Received Power，参考信号接收功率）和SINR（Signal to Interference Plus Noise Ratio，信号干扰噪声比）信息。（等待上报无线覆盖等级和小区ID，超时时间为3分钟左右）

- 查询成功：则根据如下方式计算可同时升级的并发数计算，并按照**步骤10**进行处理。
  - 如下图所示，如果设备的RSRP强度和SINR强度均落在等级“0”中，则同时可以对该小区的50个相同信号覆盖区间的设备进行同时升级。
  - 如果设备的RSRP强度和SINR强度分别落在等级“0”和“1”中，则以信号较弱的等级“1”为准，则只能同时对该小区的10个设备进行升级。
  - 如果设备的RSRP强度和SINR强度分别落在等级“1”和“2”中，则以信号较弱的等级“2”为准，则只能同时对该小区的1个设备进行升级。
  - 如果设备的RSRP强度和SINR强度不在该3个等级范围内，且均可以查询到，则按照信号最弱覆盖等级“2”处理，则只能同时对1个设备进行升级。

无线覆盖等级	RSRP门限范围 (dBm)	SINR门限范围 (dB)
0	-105 <= RSRP	7 <= SINR
1	-115 <= RSRP < -105	-3 <= SINR < 7
2	-125 <= RSRP < -115	-8 <= SINR < -3

说明

如果用户在软件升级中发现同时进行升级的设备数较少，则可以联系当地运营商检查和优化设备所在小区的无线覆盖情况。

- 查询失败：则按照流程**9**进行处理。

9. 物联网平台继续下发查询小区ID信息的命令，获取终端设备所在的小区ID信息。

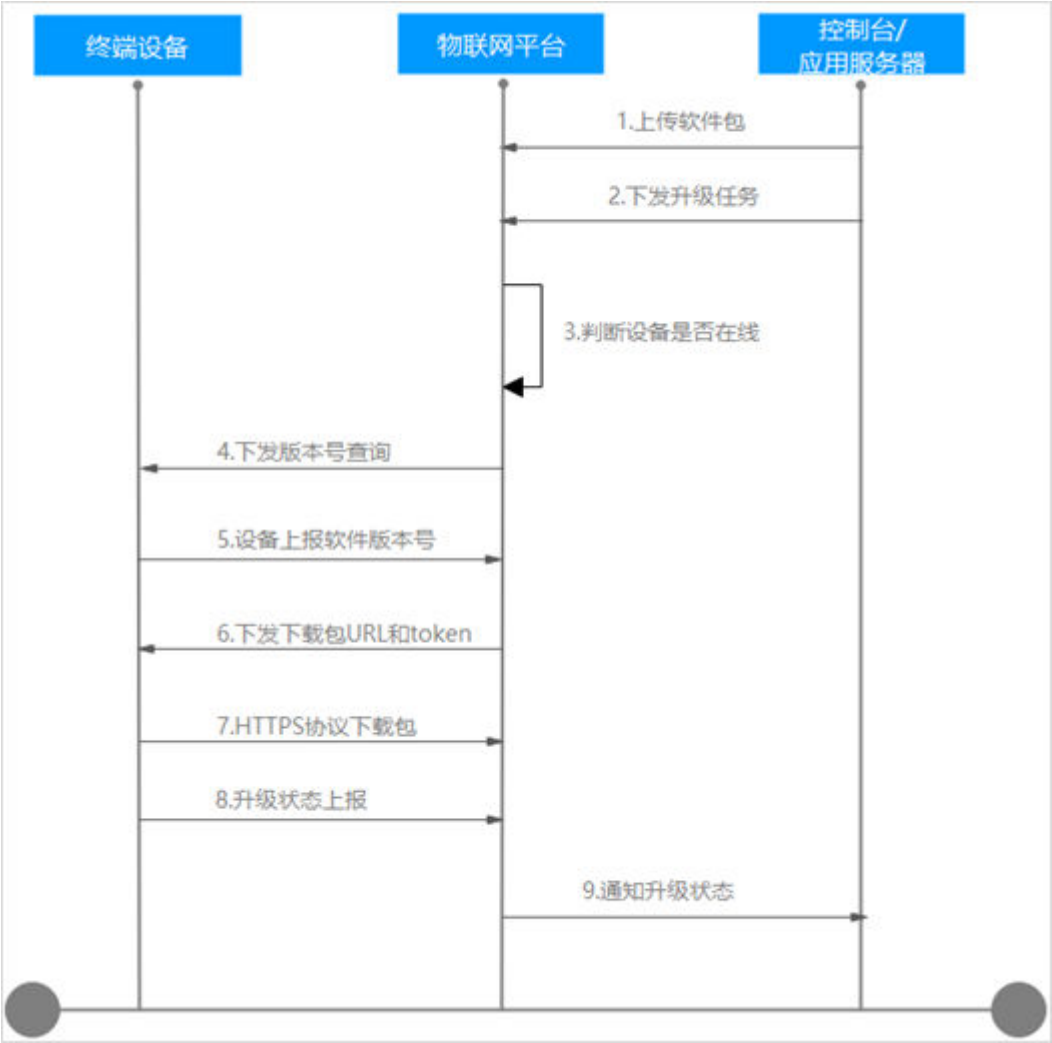
- 如果查询成功：物联网平台支持同时对该小区的10个相同情况的设备进行软件升级。
- 如果查询失败：则升级失败。

10～12. 物联网平台通知设备有新的软件包版本，设备启动软件包的下载。软件包的下载按照分片的方式进行下载，支持断点续传功能，通过软件包分片中携带的“versionCheckCode”确定是否属于同一个软件包。下载完成后，设备知会物联网平台软件包已下载完毕。（第11步超时时间为60分钟）

13～14. 物联网平台向设备下发升级的命令，终端设备进行升级操作，升级完成后终端设备向物联网平台反馈升级的结果。（等待设备上报升级结果和升级状态，超时时间为30分钟）

15. 物联网平台向控制台/应用服务器通知升级的结果。

MQTT 协议设备软件升级流程



MQTT协议SOTA升级流程的详细说明：

1～2. 用户在设备管理服务的控制台上传软件包，并在控制台或者应用服务器上创建软件升级任务。

3. 平台感知设备是否在线，当设备在线时立即触发升级协商流程。当设备不在线时，等待设备上线订阅升级Topic，平台感知设备上线，触发升级协商流程。（等待设备上线时间25小时以内）

4~5. 平台向设备下发查询设备软件版本号的命令，查询成功后，物联网平台根据升级的目标版本判断设备是否需要升级 。（第5步超时时间3分钟）

- 如果返回的软件版本信息与升级的目标版本信息相同，则升级流程结束，不做升级处理，升级任务置为成功。
- 如果返回的软件版本信息与升级的目标版本信息不同，且该版本号支持升级，则继续进行下一步的升级处理。

6~7. 物联网平台下发下载包URL**访问这里**，token及包的相关信息，用户根据下载包URL和token通过HTTPS协议来下载软件包，24小时后token无效。（下载包和升级状态上报超时时间为24小时）

8. 终端设备进行下载包升级操作，升级完成后终端设备向物联网平台反馈升级的结果。(设备升级完成后返回的版本号和设置的版本一致为成功)

9. 物联网平台向控制台/应用服务器通知升级的结果。

上传软件包

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择“设备 > 软固件升级”。

步骤3 选择“软件列表”页签，单击“上传软件”。

步骤4 在弹出的页面中，按照如下信息填写参数后，单击“确认”上传软件包。

在上传软件包时，需要填写如下信息：

参数名称	说明
软件文件	加载软件包，软件包名不能带中文字符。
软件版本	软件包的版本。（设备升级完成后需要上报升级版本号，平台检查设备上报的版本号是否与此参数一致，一致为升级成功。）

参数名称	说明
选择产品	选择对应设备的**产品模型**。
支持设备源版本	手动输入，如需输入多个，可以在输入完一个版本后，单击“回车”按键，再输入下一个。
软件包分片大小	终端下载软件包的每个分片的大小，单位为byte。取值范围：32~500，默认值500。

说明

目前平台仍然支持上传已签名的软件包，但无法修改公钥。如果您上传的是签过名的软件包，平台会使用之前上传的公钥对签名进行校验。如果您之前没有上传过公钥或者签名的私钥发生变化，签名会校验失败从而导致软件包上传失败。

后续平台将下线上传签名软件包功能，为了保障您能够正常使用软件升级功能，请直接上传需要下发给设备的文件。

----结束

批量设备软件升级

用户对批量设备进行软件升级有两种方式：

1. 应用服务器通过调用的“创建软件升级任务”API接口，创建批量设备的升级任务，详情请参考**创建批量任务**。
2. 通过控制台，创建批量设备的软件升级任务。

下面将重点介绍通过控制台创建批量设备的软件升级任务。

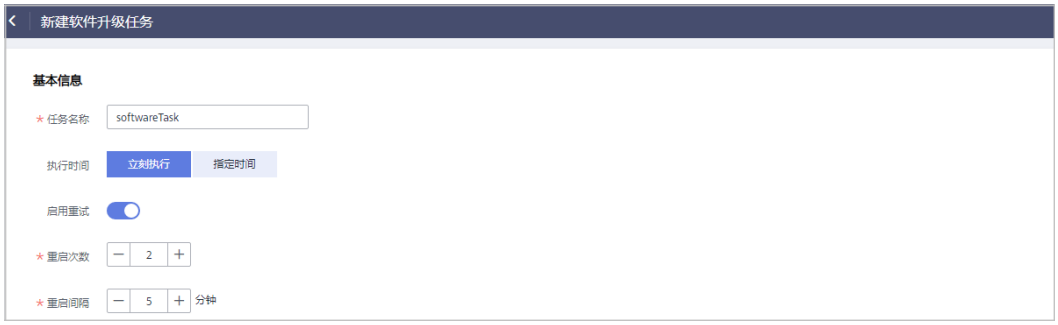
步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择"设备 > 软固件升级"，单击“升级任务”。

步骤3 选择“软件升级”页签，单击“新建任务”按钮，进入新建软件升级任务页面。

步骤4 设置“任务信息”，填写任务名称、执行时机、启用重试。

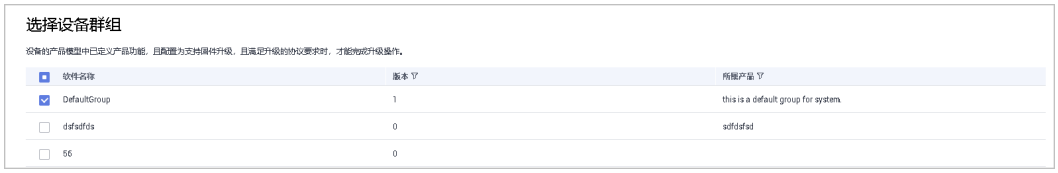
启用重试后，可以设置重启次数和重启间隔。重启次数建议设置为2次，重启间隔设置为5分钟，即设备升级失败后，隔5分钟后会进行升级重试。



步骤5 选择需要升级的软件包。

步骤6 选择需要升级的设备或者设备群组，然后单击“立即创建任务”。

设备群组可以参考**群组与标签**创建需要升级的设备群组，并绑定对应的设备。



步骤7 创建完批量升级任务后，可以在软件升级任务列表中查看批量任务的执行结果。单击对应任务"查看"按钮，可以在“执行详情”界面查看每个设备的升级结果。

说明

如果升级任务正在执行中，是不允许删除任务的，如需删除，请先在任务列表中，手动停止任务后，再删除升级任务。

----结束

常见问题

软/固件升级业务热点咨询问题如下，更多咨询问题请访问**查看更多**。

- **目标版本可以比当前版本低吗?**
- **软/固件包及其版本号如何获取?**
- **在软/固件升级任务中，业务处理是否会中断?**
- **为什么创建了软固件升级任务后，任务立即结束?**
- **常见的软/固件升级错误有哪些?**

相关 API 接口

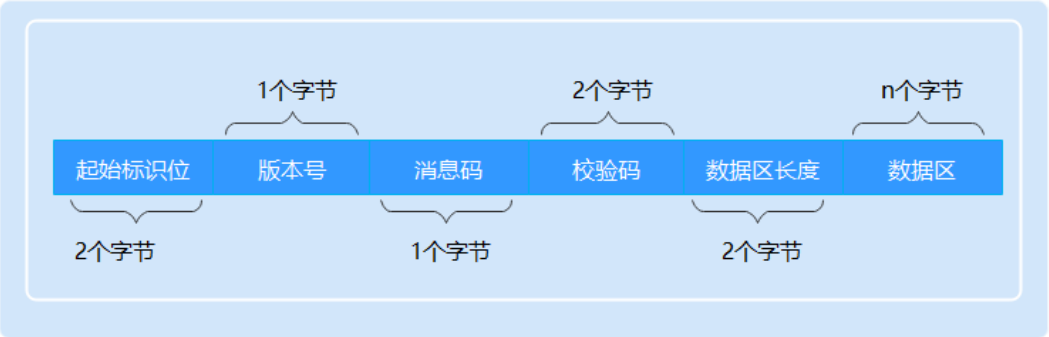
- **创建批量任务**
- **查询批量任务列表**
- **查询批量任务**

3.8.2 设备侧适配开发指导

概述

设备的OTA软件升级是基于华为定义的**PCP协议**进行的，设备侧需根据PCP协议定义的交互流程进行适配开发。下面我们将结合物联网平台与设备的软件升级交互流程，介绍终端设备将如何基于PCP协议构建交互过程中的请求消息和应答消息，帮助您更好的根据PCP协议进行终端侧的软件升级功能开发。

下面我们先了解下PCP消息的结构，PCP协议的请求消息和应答消息都遵循相同的消息结构，主要由这几部分组成：



PCP协议消息由：起始标识位、版本号、消息码、校验码、数据区长度和数据区组成，各字段的要求和描述如下表所示。

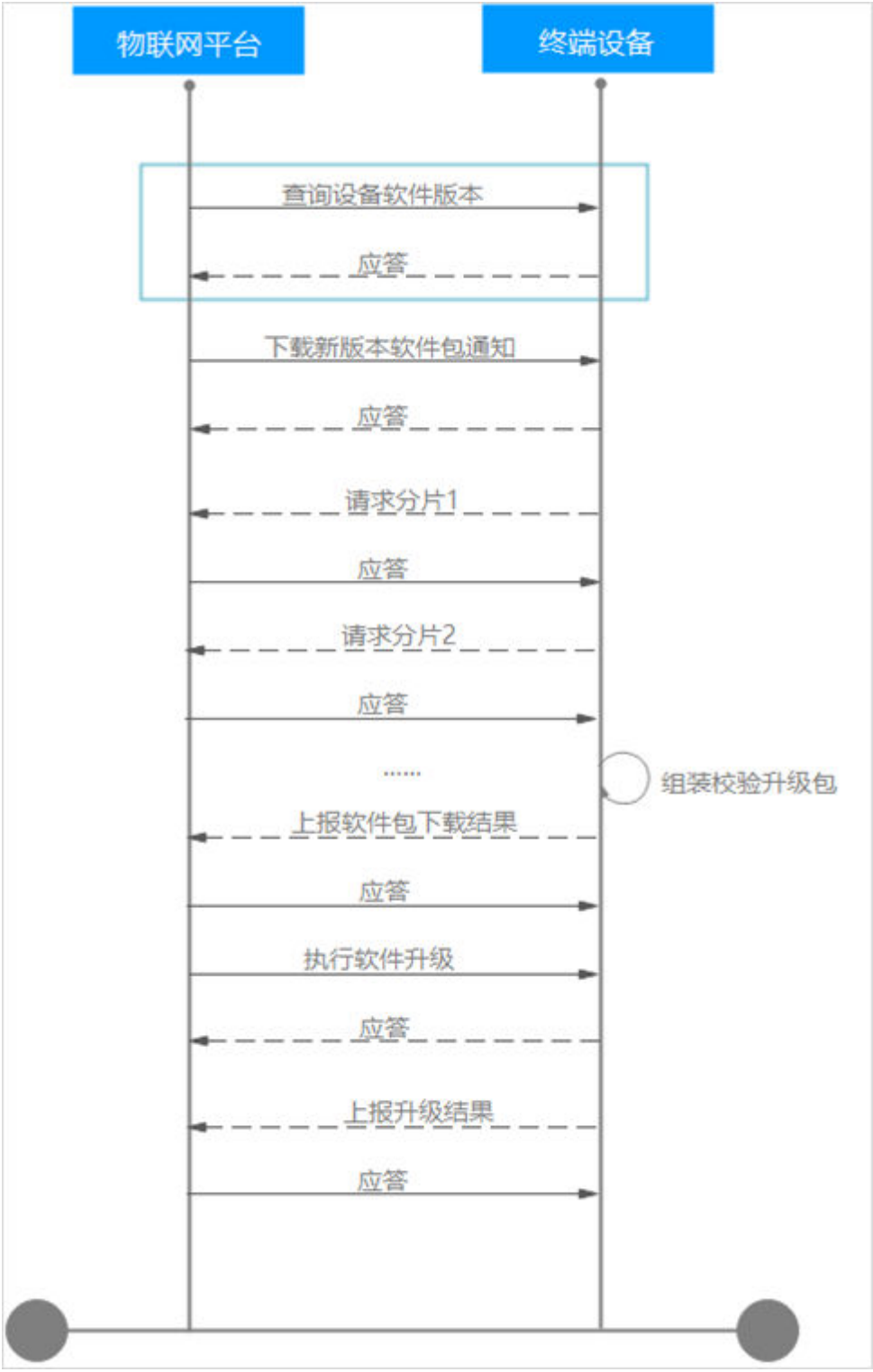
字段名	字段类型	描述和要求
起始标识	WORD	起始标识，固定为0XFFFE。
版本号	BYTE	高四位预留；低四位表示协议版本号，当前为1。
消息码	BYTE	标识物联网平台与设备之间的请求消息类型，应答消息的消息码和请求消息相同。消息码的定义为： • 0-18：预留消息码，暂未使用。 • 19：查询设备版本。 • 20：下载新版本软件包通知。 • 21：请求下载升级包。 • 22：上报升级包下载结果。 • 23：执行软件升级。 • 24：上报升级结果。 • 25-127：预留消息码，暂未使用。
校验码	WORD	从起始标识到数据区的最后一个字节的CRC16校验值，计算前先把校验码字段置为0，计算完成后把结果写到校验码字段。 **说明** CRC16算法：CRC16/CCITT x16+x12+x5+1
数据区长度	WORD	数据区的长度。
数据区	BYTE[n]	可变长度，具体由各个指令定义，可参考下面介绍的各个指令对应的请求消息和应答消息定义。

数据类型	描述
BYTE	无符号一字节整数

数据类型	描述
WORD	无符号二字节整数
DWORD	无符号四字节整数
BYTE[n]	n字节的十六进制数
STRING	字符串

查询设备版本号

从设备的软件升级流程（本流程只描述物联网平台与设备基于PCP协议交互的流程，完整的软件升级流程请参考**LwM2M协议设备软件升级流程**）可以看到，首先物联网平台向设备下发查询版本号信息，设备进行应答。



物联网平台发送消息

根据**PCP消息结构的定义**可以得出，物联网平台向设备下发查询版本号时，各消息字段的填写如下：

- 起始标识：固定为消息流的前2个字节，固定为FFFE。
- 版本号：数据类型为1个字节整数，且固定为1，即在消息流中为01。

- 消息码：数据类型为1个字节整数，查询设备版本的消息码为19，转换为十六进制为13。
- 校验码：数据类型为2个字节整数，先将校验码置为0000，然后将完整的消息码流进行CRC16的算法计算得到校验码，再将得到的校验码替换原消息中的0000。
- 数据区长度：数据类型为2个字节整数，代表数据区的消息长度，根据数据区的数据结构可以得出该条消息无数据区，即数据区长度为0000。
- 数据区：数据区代表要真正发送给设备的数据，根据查询版本信息的数据区定义，该条消息是没有实际要传送的数据的，即无需数据区字段。

字段	数据类型	描述及要求
无数据区		

因此将查询版本消息的码流组合起来得到：FFFE 01 13 0000 0000。前面的校验码时讲了，需要将组合后的消息码流进行CRC16算法（物联网平台提供了基于JAVA和C语言的**CRC16算法代码样例**，您可以直接使用）得到校验码4C9A，然后将该校验码替换原码流中的0000后得到FFFE01134C9A0000，该消息码流即为物联网平台发送给设备的查询版本信息的消息码流。

设备返回的应答消息

设备收到物联网平台要查询设备的软件版本号消息，设备要向物联网平台反馈查询的结果，各消息字段的填写如下。

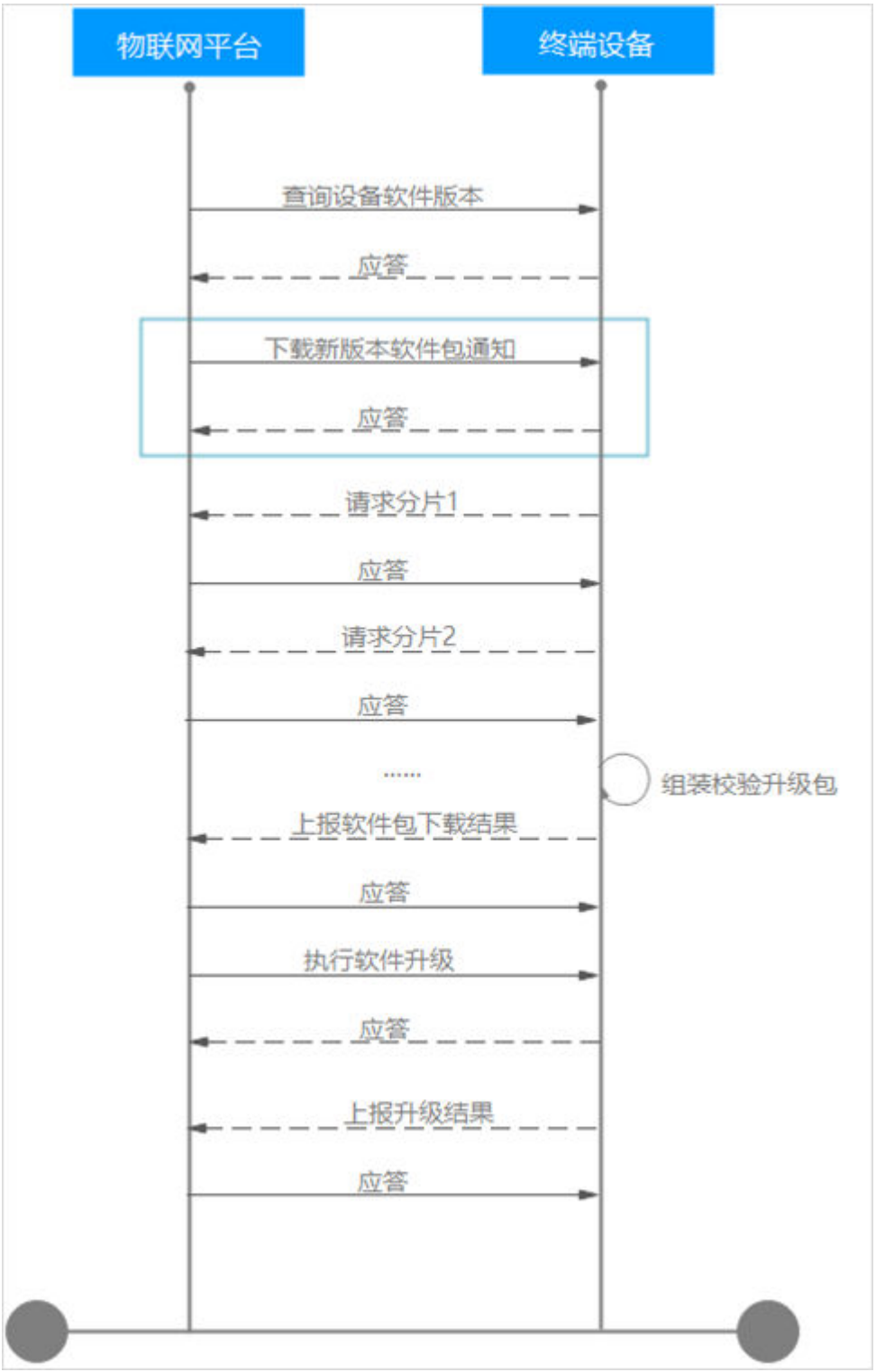
- 起始标识固定为：FFFE。
- 版本号固定为：01。
- 消息码：与请求的消息码一致，为13。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段的数据类型得出数据区长度为17个字节，转换为十六进制为：0011。
- 数据区：根据数据区的定义可知，处理成功的结果码为00，版本号信息假设为V0.9，将V0.9进行ASCII转码得到56302E39，由于版本号的数据类型为BYTE[16]，即16个字节，当前只有4个字节，因此需要在版本号数据后面补0，得到56302E390000000000000000000000000。因此，数据区合并后为0056302E390000000000000000000000000。

字段	数据类型	描述及要求
结果码	BYTE	“0X00”处理成功
当前版本号	BYTE[16]	当前版本号，由ASCII字符组成，位数不足时，后补“0X00”。

将查询版本信息的消息流组合起来得到：FFFE 01 13 0000 0011 0056302E390000000000000000000000000。前面讲到，还要将消息流进行CRC16算法计算得到校验码为8DE3。因此，物联网平台向设备查询版本号信息，设备向平台返回的消息流为FFFE01138DE300110056302E390000000000000000000000000。

下载新版本软件包通知

根据PCP协议约定的交互流程，查询完版本号后，物联网平台下发指令让设备下载新版本的软件包。



物联网平台发送消息

根据**PCP消息结构的定义**可以得出，物联网平台向设备下发下载新版本软件包通知时，各消息字段的填写如下。

- 起始标识固定为：FFFE。
- 版本号固定为：01。
- 消息码：此处为新版本通知，查询**消息码**表可以知道新版本通知为20，转换为十六进制为14。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的消息字段可以得出，数据区长度为22个字节，转换为十六进制为：0016。
- 数据区：根据数据区的定义可知。
  - 目标版本号：由16个字节组成，假设升级的目标版本号为v1.0版本，转换为十六进制并在后面14个字节补充0后得到：56312E30000000000000000000000000。
  - 升级包分片大小：由2个字节组成，单位为byte，用户**上传软件包**时可以手动输入升级包分片大小，如果不设置默认为500byte，大小为32~500之间。假设为500byte，转换为十六进制后为：01F4。
  - 升级包分片总数：由2个字节组成，由软件包大小除以每个分片的大小并向上取整获得。假设软件包大小为500byte，则分片数量为1，转换为十六进制后为：0001。
  - 检验码：由2个字节组成，目前已废弃，固定为：0000。

字段	数据类型	描述及要求
目的版本号	BYTE[16]	目的版本号，由ASCII字符组成，位数不足时，后补“0X00”。
升级包分片大小	WORD	每个分片的大小
升级包分片总数	WORD	升级包分片总数
升级包校验码	WORD	固定为：0000

将下载新版本软件包通知的消息流组合起来得到：FFFE 01 14 0000 0016 56312E30000000000000000000000000 01F4 0001 0000。前面说了，还要将消息流进行CRC16算法计算得到校验码为02F7。因此，物联网平台向设备通知下载新版本软件包的信息，物联网平台向设备发送的消息流为FFFE011402F7001656312E30000000000000000000000000001F400010000。

设备返回的应答消息

设备收到下载新版本软件包通知后，设备向物联网平台返回应答消息，是否允许设备进行升级，各消息字段的填写如下。

- 起始标识固定为：FFFE。
- 版本号固定为：01。
- 消息码：与请求的消息码一致，为14。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段的数据类型得出数据区长度为1个字节，转换为十六进制为：0001。

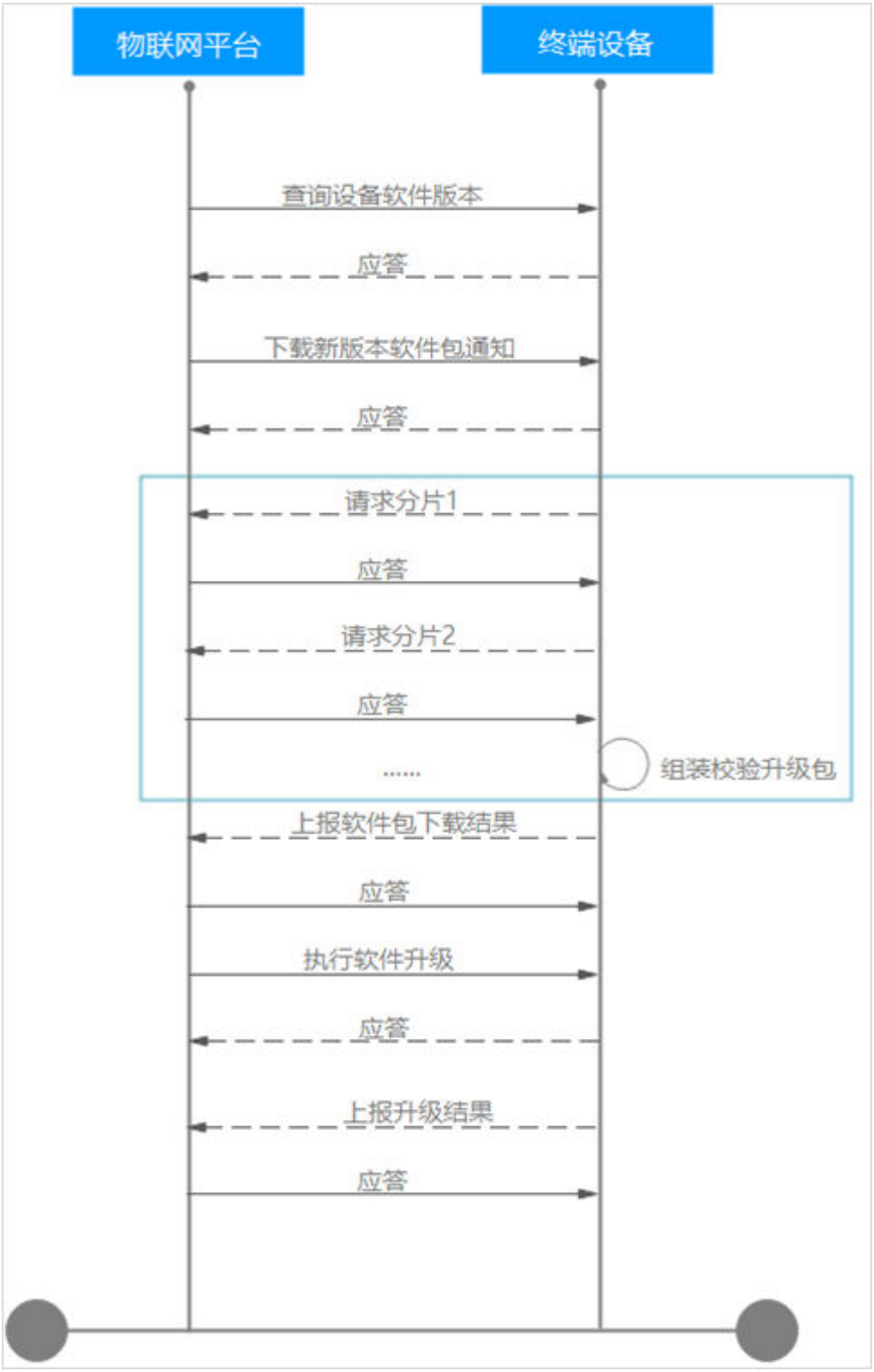
- 数据区：设备根据自身的情况对平台下发的新版本通知进行响应，本示例以设备应答“允许升级”为例进行介绍，得出数据区为：00。其它应答消息请根据应答消息字段进行适配。

字段	数据类型	描述及要求
结果码	BYTE	“0X00”允许升级 “0X01”设备使用中 “0X02”信号质量差 “0X03”已经是最新版本 “0X04”电量不足 “0X05”剩余空间不足 “0X09”内存不足 “0X7F”内部异常

将设备给物联网平台的应答消息流组合起来得到：FFFE 01 14 0000 0001 00。前面说了，还要将消息流进行CRC16算法计算得到校验码为D768。因此，设备向平台返回的应答消息流为FFFE0114D768000100。

下载升级包

根据PCP协议约定的交互流程，物联网平台通知设备有新的软件版本时，设备向物联网平台请求下载软件包，按照分片的序号进行下载。



设备发送的请求消息

根据**PCP消息结构的定义**可以得出，设备向物联网平台发送的请求软件包分片的第一条消息，各消息字段的填写如下。

- 起始标识固定为：FFFE。
- 版本号固定为：01。

- 消息码：查询**消息码**表可知请求升级包的消息码为21，转换为十六进制为：15。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段的数据类型得出数据区长度为18个字节，转换为十六进制为：0012。
- 数据区：目标版本号为平台下发的新版本通知版本号，即v1.0，转换为十六进制为56312E300000000000000000000000000，分片序号为第0个分片，即0000。

字段	数据类型	描述及要求
目的版本号	BYTE[16]	目的版本号，由ASCII字符组成，位数不足时，后补“0X00”。
分片序号	WORD	表示请求获取的分片序号，从0开始计算，分片的总数为软件包大小除以每个分片的大小并向上取整获得。设备可以保存已经收到的分片，下次直接从缺失的分片开始请求，达到断点续传的效果。

设备向物联网平台发送请求软件包分片的第一条消息为：FFFE 01 15 0000 0012 56312E300000000000000000000000000 0000（CRC16校验前），经CRC16计算得到校验码为：5618。则替换校验码后设备发送的第一条请求分片消息为：FFFE01155618001256312E300000000000000000000000000000000。

其它分片请求的消息流只需要替换分片序号后，重新计算并替换CRC16校验码即可，此处就不再展开。

物联网平台的应答消息

物联网平台收到设备的请求软件包分片消息后，将会给设备下发分片的数据。物联网平台向设备响应的第一条请求分片的消息，各消息字段的填写如下。

- 起始标识固定为：FFFE。
- 版本号固定为：01。
- 消息码：与请求的消息码一致：15。
- 校验码：CRC16计算前先用0000替代。
- 数据区：先讲数据区再讲数据区长度。结果码：00，分片序号：第0个分片：0000，分片数据：跟软件包定义的内容有关，我们假设软件包内容为HELLO, IoT SOTA!，经ASCII码转换为十六进制为：48454C4C4F2C20496F5420534F544121，共16字节。用户**上传软件包**时手动输入升级包分片大小为500byte，即最大长度为500字节。这种情况下，无需在数据的后面补充0。

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X80升级任务不存在。 0X81指定的分片不存在。
分片序号	WORD	表示返回的分片序号。
分片数据	BYTE[n]	分片的内容，n为实际的分片大小。如果结果码不为0，则不带此字段。

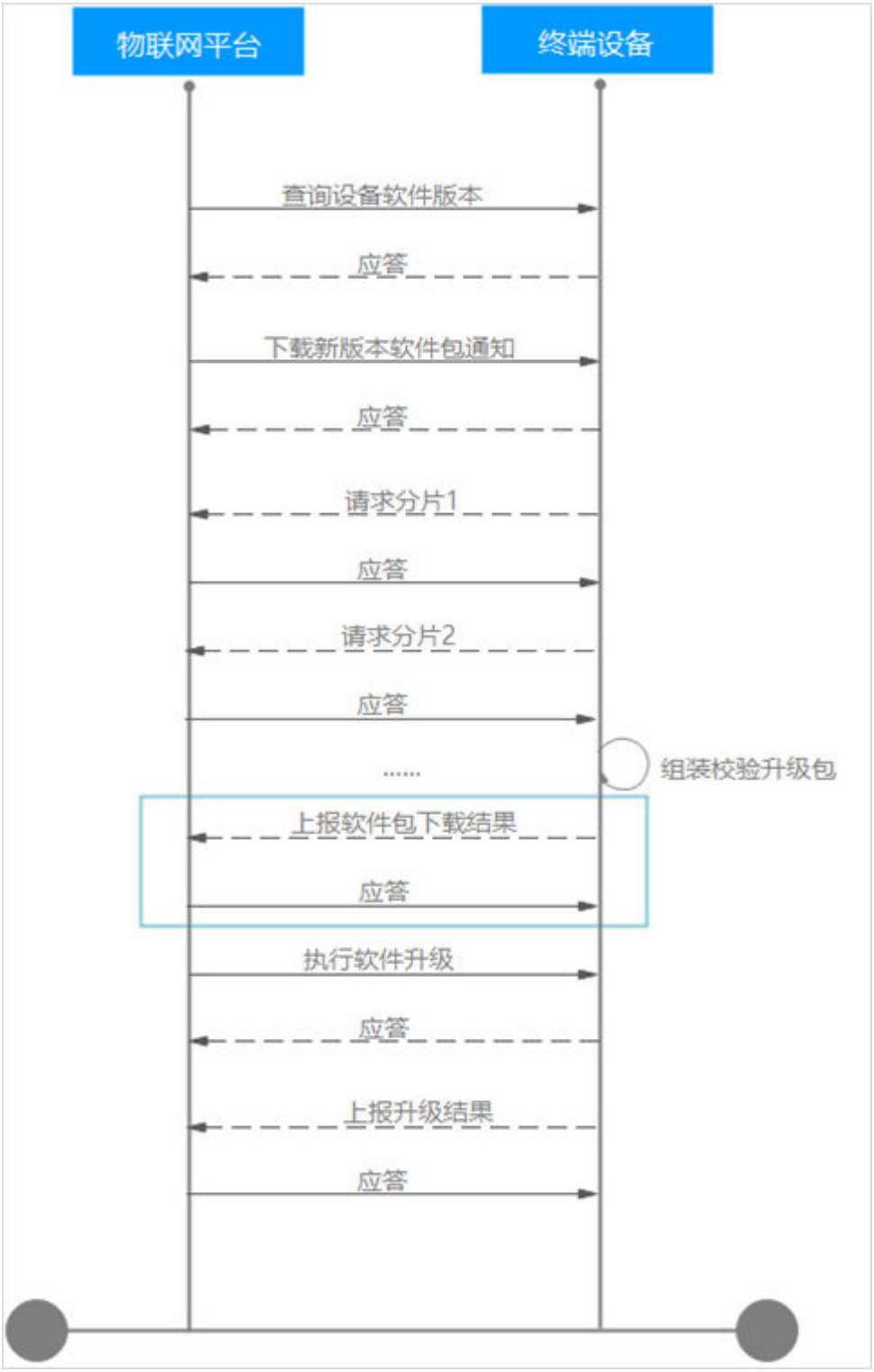
- 数据区长度：根据数据区的字段定义得出该数据长度为19，转换为十六进制为：0013。

物联网平台向设备发送的第一个软件包分片消息为：FFFE 01 15 0000 0013 00 0000 48454C4C4F2C20496F5420534F544121（CRC16校验前），经CRC16计算得到校验码为：E107。则替换校验码后物联网平台向设备发送的第一个软件包分片消息为：FFFE0115E10700130000048454C4C4F2C20496F5420534F544121。

其它软件包分片的消息流只需要替换分片序号和分片数据后，重新计算并替换CRC16校验码即可，此处就不再展开。

上报下载结果

根据PCP协议约定的交互流程，设备接收完所有分片数据并组装完软件包后，需要向物联网平台上报软件包的下载结果。



设备发送的请求消息

根据**PCP消息结构的定义**可以得出，设备向物联网平台发送的上报软件包下载结果消息，各个消息字段的填写如下：

- 起始标识固定为：FFFE。
- 版本号固定为：01。

- 消息码：与请求的消息码一致，为16。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段的数据类型得出数据区长度为1个字节，转换为十六进制为：0001。
- 数据区：上报软件包的下载结果，比如下载成功，设备侧上报00。

字段	数据类型	描述及要求
下载状态	BYTE	0X00下载成功。 0X05剩余空间不足。 0X06下载超时。 0X07升级包校验失败。 0X08升级包类型不支持。

设备向物联网平台发送升级包下载结果的消息为：FFFE 01 16 0000 0001 00（CRC16校验前），经CRC16计算得到校验码为：850E。则替换校验码后设备发送的升级包下载结果的消息为：FFFE0116850E000100。

物联网平台的应答消息

物联网平台收到设备上报的软件包下载结果后，将会向设备返回应答消息，各个消息字段的填写如下。

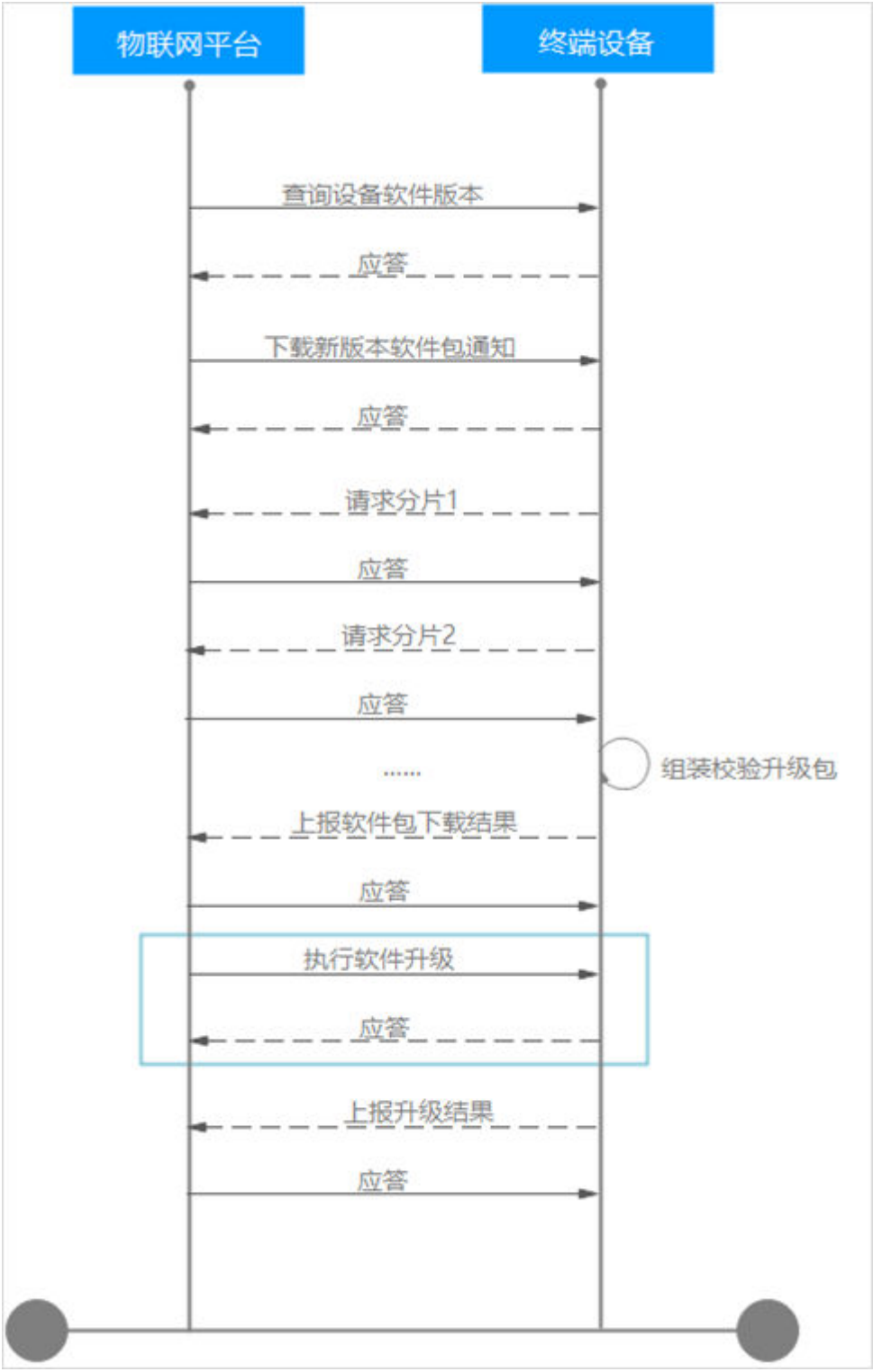
- 起始标识固定为：FFFE。
- 版本号固定为：01。
- 消息码：与请求的消息码一致：16。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段定义得出该数据长度为1个字节，转换为十六进制为：0001。
- 数据区：处理成功，则返回00，处理失败返回80。本示例以返回00处理成功为例进行说明。

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X80升级任务不存在。

物联网平台向设备应答的消息为：FFFE 01 16 0000 0001 00 （CRC16校验前），经CRC16计算得到校验码为：850E。则替换校验码后物联网平台向设备应答的消息为：FFFE0116850E000100。

执行软件升级

根据PCP协议约定的交互流程，物联网平台收到设备发送的软件包下载结果通知后，需要通知设备进行升级操作。



物联网平台发送的请求消息

根据**PCP消息结构的定义**可以得出，物联网平台向设备发送执行软件升级消息，各个消息字段的填写如下：

- 起始标识固定为：FFFE。
- 版本号固定为：01。

- 消息码：与请求的消息码一致，为17。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段的数据类型得出无数据区，即为0字节，转换为十六进制为：0000。
- 数据区：无数据区，无需携带该字段。

字段	数据类型	描述及要求
无数据区		

物联网平台向设备下发的执行软件升级的消息为：FFFE 01 17 0000 0000（CRC16校验前），经CRC16计算得到校验码为：CF90。则替换校验码后物联网平台向设备发送的消息为：FFFE0117CF900000。

设备发送的应答消息

设备收到物联网平台下发的执行升级消息后，将对收到消息后的执行动作进行应答，各消息字段的填写如下。

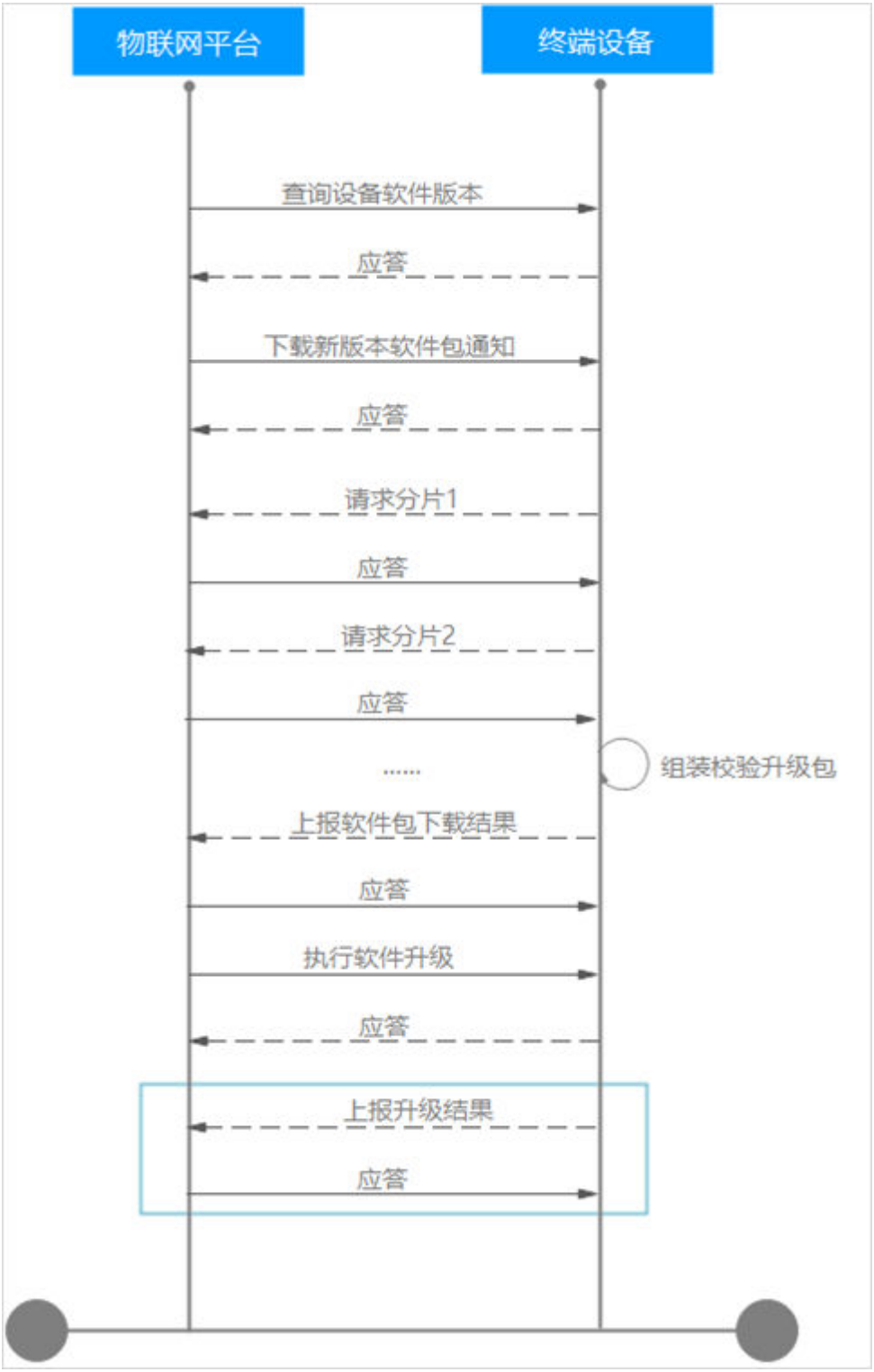
- 起始标识固定为：FFFE。
- 版本号固定为：01。
- 消息码：与请求的消息码一致：17。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段定义得出该数据长度为1个字节，转换为十六进制为：0001。
- 数据区：处理成功，则返回00，其它处理结果请参考数据区定义。本示例以返回00处理成功为例进行说明。

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X01设备使用中。 0X04电量不足。 0X05剩余空间不足。 0X09内存不足。

设备向物联网平台应答的消息为：FFFE 01 17 0000 0001 00 （CRC16校验前），经CRC16计算得到校验码为：B725。则替换校验码后设备返回的响应消息为：FFFE0117B725000100 。

上报升级结果

根据PCP协议约定的交互流程，设备在执行完软件升级后，将会向物联网平台上报升级的结果。



设备发送的请求消息

根据**PCP消息结构的定义**可以得出，设备向物联网平台上报升级结果，各个消息字段的填写如下：

- 起始标识固定为：FFFE。
- 版本号固定为：01。

- 消息码：与请求的消息码一致，为18。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段的数据类型得出数据区长度为17字节，转换为十六进制为：0011。
- 数据区：结果码，以上报升级成功为例，结果码为00。当前版本号：升级完成后的版本号，与物联网平台下发的软件版本一致，即v1.0，转换为十六进制为：56312E300000000000000000000000000。

字段	数据类型	描述及要求
结果码	BYTE	0X00升级成功。 0X01设备使用中。 0X04电量不足。 0X05剩余空间不足。 0X09内存不足。 0X0A安装升级包失败。 0X7F内部异常。
当前版本号	BYTE[16]	设备当前版本号。

设备向物联网平台上报升级结果的消息为：FFFE 01 18 0000 0011 0056312E300000000000000000000000000（CRC16校验前），经CRC16计算得到校验码为：C7D2。则替换校验码后设备向物联网平台上报升级结果码流为：FFFE0118C7D200110056312E300000000000000000000000000。

物联网平台发送的应答消息

物联网平台收到设备上报的升级结果消息后，将对设备进行应答，各个消息字段的填写如下。

- 起始标识固定为：FFFE。
- 版本号固定为：01。
- 消息码：与请求的消息码一致：18。
- 校验码：CRC16计算前先用0000替代。
- 数据区长度：根据数据区的字段定义得出该数据长度为1个字节，转换为十六进制为：0001。
- 数据区：处理成功，则返回00，升级任务不存在80。本示例以返回00处理成功为例进行说明。

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X80升级任务不存在。

物联网平台向设备的应答消息为：FFFE 01 18 0000 0001 00（CRC16校验前），经CRC16计算得到校验码为：AFA1。则替换校验码后物联网平台返回的应答消息为：FFFE0118AFA1000100 。

至此，设备的软件升级功能适配就完成了，赶紧动手试一试吧。

CRC16 算法代码样例

基于JAVA的CRC16算法样例：

```
public class CRC16 {

    /*
     * CCITT标准CRC16(1021)余数表 CRC16-CCITT ISO HDLC, ITU X.25, x16+x12+x5+1 多项式
     * 高位在先时生成多项式 Gm=0x11021 低位在先时生成多项式，Gm=0x8408 本例采用高位在先
     */
    private static int[] crc16_ccitt_table = { 0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
            0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef, 0x1231, 0x0210, 0x3273, 0x2252,
            0x52b5, 0x4294, 0x72f7, 0x62d6, 0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
            0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a, 0xb54b, 0x8528, 0x9509,
            0xe5ee, 0xf5cf, 0xc5ac, 0xd58d, 0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
            0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5, 0x6886, 0x78a7,
            0x0840, 0x1861, 0x2802, 0x3823, 0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
            0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12, 0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e,
            0x9b79, 0x8b58, 0xbb3b, 0xab1a, 0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
            0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97, 0x6eb6, 0x5ed5, 0x4ef4,
            0x3e13, 0x2e32, 0x1e51, 0x0e70, 0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
            0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1, 0x30c2, 0x20e3,
            0x5004, 0x4025, 0x7046, 0x6067, 0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
            0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256, 0xb5ea, 0xa5cb, 0x95a8, 0x8589,
            0xf56e, 0xe54f, 0xd52c, 0xc50d, 0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
            0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3, 0x36f2, 0x0691, 0x16b0,
            0x6657, 0x7676, 0x4615, 0x5634, 0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
            0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e,
            0x8bf9, 0x9bd8, 0xabbb, 0xbb9a, 0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
            0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9, 0x7c26, 0x6c07, 0x5c64, 0x4c45,
            0x3ca2, 0x2c83, 0x1ce0, 0x0cc1, 0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
            0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0 };

    /**
     *
     * @param reg_init
     *            CRC校验时初值
     * @param message
     *            校验值
     * @return
     */
    private static int do_crc(int reg_init, byte[] message) {
        int crc_reg = reg_init;
        for (int i = 0; i < message.length; i++) {
            crc_reg = (crc_reg >> 8) ^ crc16_ccitt_table[(crc_reg ^ message[i]) & 0xff];
        }
        return crc_reg;
    }

    /**
     * 根据数据生成CRC校验码
     *
     * @param message
     *            byte数据
     *
     * @return int 返校验码
     */
    public static int do_crc(byte[] message) {
        // 计算CRC校验时初值从0x0000开始。
        int crc_reg = 0x0000;
        return do_crc(crc_reg, message);
    }
}
```

基于C的CRC16算法样例：

```
/**
* CCITT标准CRC16(1021)余数表 CRC16-CCITT ISO HDLC, ITU X.25, x16+x12+x5+1 多项式
* 高位在先时生成多项式 Gm=0x11021 低位在先时生成多项式，Gm=0x8408 本例采用高位在先
```

```
*/
const unsigned short crc16_table[256] = {
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
    0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
    0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
    0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
    0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
    0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
    0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
    0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
    0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
    0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
    0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
    0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
    0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
    0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
    0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
    0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
    0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
    0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
    0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
    0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
    0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
    0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
    0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
};

int do_crc(int reg_init, byte* data, int length)
{
    int cnt;
    int crc_reg = reg_init;
    for (cnt = 0; cnt < length; cnt++)
    {
        crc_reg = (crc_reg >> 8) ^ crc16_table[(crc_reg ^ *(data++)) & 0xFF];
    }
    return crc_reg;
}

int main(int argc, char **argv)
{
    //FFFE011300000000用byte数组表示：
    byte message[8] = {0xFF,0xFE,0x01,0x13,0x00,0x00,0x00,0x00};
    // 计算CRC校验时初值从0x0000开始
    int a = do_crc(0x0000, message, 8);
    printf("a ==> %x\n", a);
}
```

3.8.3 PCP 协议介绍

平台升级协议（PCP协议）规定了设备和平台之间升级的通信内容与格式。

本协议规定设备和IoT平台（以下简称“平台”）之间的应用层升级协议（简称“PCP协议”），用于实现设备的升级。

通讯方式

1. PCP协议运行在应用层，底层可以是LwM2M/CoAP/MQTT或者其他非流式协议。

2. 由于PCP协议消息没有使用单独的端口号，并且不依赖于底层协议，为了和设备业务消息区分，PCP协议固定以0XFFFE作为起始字节。因此要求设备的业务消息的前两个字节不能是0XFFFE，更多细节参考附录**PCP消息识别**。
3. 本协议消息采用一问一答模式，所有请求消息都有一个响应消息。

消息结构

字段名	字段类型	描述和要求
起始标识	WORD	起始标识，固定为0XFFFE。
版本号	BYTE	高四位预留；低四位表示协议版本号，当前为1。
消息码	BYTE	标识物联网平台与设备之间的请求消息类型，应答消息的消息码和请求消息相同。消息码的定义为： ● 0-18：预留消息码，暂未使用。 ● 19：查询设备版本。 ● 20：下载新版本软件包通知。 ● 21：请求下载升级包。 ● 22：上报升级包下载结果。 ● 23：执行软件升级。 ● 24：上报升级结果。 ● 25-127：预留消息码，暂未使用。
校验码	WORD	从起始标识到数据区的最后一个字节的CRC16校验值，计算前先把校验码字段置为0，计算完成后把结果写到校验码字段。 **说明** CRC16算法：CRC16/CCITT x16+x12+x5+1
数据区长度	WORD	数据区的长度。
数据区	BYTE[n]	可变长度，具体由各个指令定义，可参考下面介绍的各个指令对应的请求消息和应答消息定义。

数据类型

数据类型	描述
BYTE	无符号一字节整数
WORD	无符号二字节整数
DWORD	无符号四字节整数
BYTE[n]	n字节的十六进制数

数据类型	描述
STRING	字符串

说明

本协议采用网络序来传输WORD和DWORD。

查询设备版本消息

请求消息：

方向：平台->设备

字段	数据类型	描述及要求
无数据区		

响应消息：

方向：设备->平台

字段	数据类型	描述及要求
结果码	BYTE	“0X00”处理成功
当前版本号	BYTE[16]	当前版本号，由ASCII字符组成，位数不足时，后补“0X00”。

说明

- 正常处理：平台根据版本号判断设备是否需要升级，如果需要，下发请求升级。
- 异常处理：如果响应超时，平台中止升级任务。

新版本通知消息

请求消息：

方向：平台->设备

字段	数据类型	描述及要求
目的版本号	BYTE[16]	目的版本号，由ASCII字符组成，位数不足时，后补“0X00”。
升级包分片大小	WORD	每个分片的大小

字段	数据类型	描述及要求
升级包分片总数	WORD	升级包分片总数
升级包校验码	WORD	固定为：0000

应答消息：

方向：设备->平台

字段	数据类型	描述及要求
结果码	BYTE	“0X00”允许升级 “0X01”设备使用中 “0X02”信号质量差 “0X03”已经是最新版本 “0X04”电量不足 “0X05”剩余空间不足 “0X09”内存不足 “0X7F”内部异常

说明

- 正常处理：如果设备不允许升级，平台中止升级任务。
- 异常处理：如果响应超时，而且没收到请求升级包消息，平台中止升级任务。

请求消息包消息

请求消息：

方向：设备->平台

字段	数据类型	描述及要求
目的版本号	BYTE[16]	目的版本号，由ASCII字符组成，位数不足时，后补“0X00”。
分片序号	WORD	表示请求获取的分片序号，从0开始计算，分片的总数为软件包大小除以每个分片的大小并向上取整获得。设备可以保存已经收到的分片，下次直接从缺失的分片开始请求，达到断点续传的效果。

响应消息：

方向：平台->设备

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X80升级任务不存在。 0X81指定的分片不存在。
分片序号	WORD	表示返回的分片序号。
分片数据	BYTE[n]	分片的内容，n为实际的分片大小。如果结果码不为0，则不带此字段。

上报升级包下载状态消息

请求消息：

方向：设备->平台

字段	数据类型	描述及要求
下载状态	BYTE	0X00下载成功。 0X05剩余空间不足。 0X06下载超时。 0X07升级包校验失败。 0X08升级包类型不支持。

响应消息：

方向：平台->设备

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X80升级任务不存在。

执行升级消息

请求消息：

方向：平台->设备

字段	数据类型	描述及要求
无数据区		

响应消息：

方向：设备->平台

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X01设备使用中。 0X04电量不足。 0X05剩余空间不足。 0X09内存不足。

上报升级结果消息

请求消息：

方向：设备->平台

字段	数据类型	描述及要求
结果码	BYTE	0X00升级成功。 0X01设备使用中。 0X04电量不足。 0X05剩余空间不足。 0X09内存不足。 0X0A安装升级包失败。 0X7F内部异常。
当前版本号	BYTE[16]	设备当前版本号。

响应消息：

方向：平台->设备

字段	数据类型	描述及要求
结果码	BYTE	0X00处理成功。 0X80升级任务不存在。

PCP 消息识别

由于PCP协议消息和设备业务消息共用一个端口和URL通讯，平台收到设备的消息时，按照如下步骤判断是PCP协议消息还是业务消息：

1. 检查设备是否支持软件升级（根据设备profile的omCapability.upgradeCapability定义），如果不支持，则认为是业务消息。
2. 检查设备软件升级协议是否是PCP，如果不是，则认为是业务消息。
3. 检查消息前两个字节是否为0XFFFE，如果不是，则认为是业务消息。
4. 检查版本号是否合法，如果不合法，则认为是业务消息。
5. 检查消息码是否合法，如果不合法，则认为是业务消息。
6. 检查校验码是否正确，如果不正确，则认为是业务消息。
7. 检查数据区长度是否正确，如果不正确，则认为是业务消息。
8. 如果以上检查都通过，认为是PCP协议消息。

说明

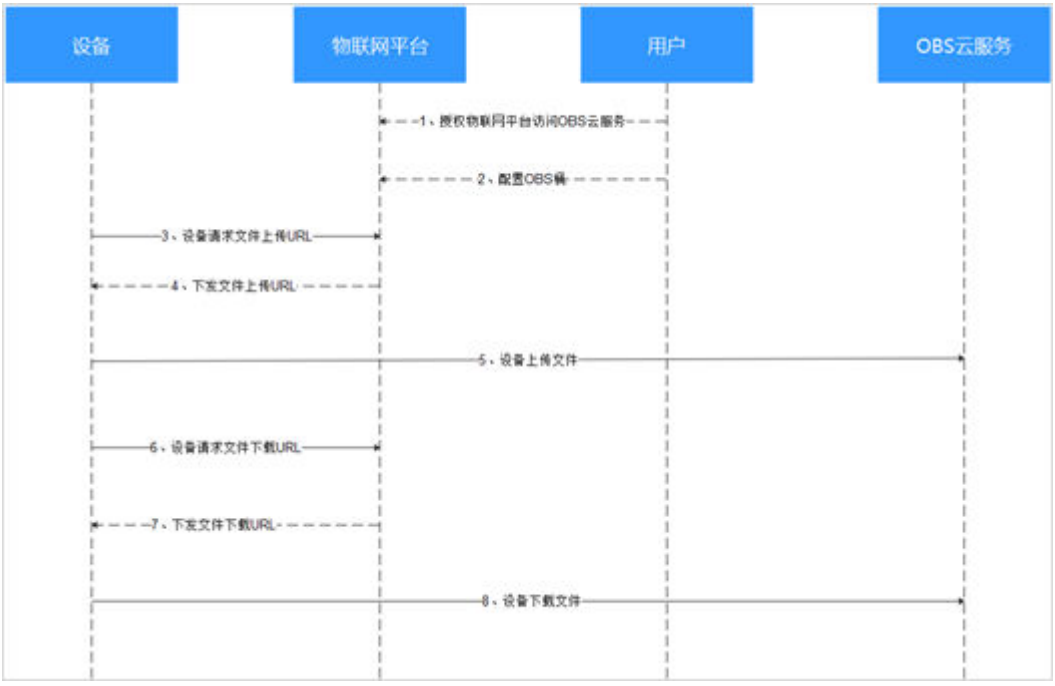
对设备的要求：需要设备保证业务消息的起始字节不是0XFFFE。

3.9 文件上传

概述

华为物联网平台支持设备将运行日志，配置信息等文件上传至平台，便于用户进行日志分析、故障定位、设备数据备份等。当设备采用HTTPS方式将文件上传到OBS服务进行备份时，您可以在OBS服务管理已上传的设备文件。

业务流程



1. 授权物联网平台访问OBS服务。

2. 配置OBS桶。

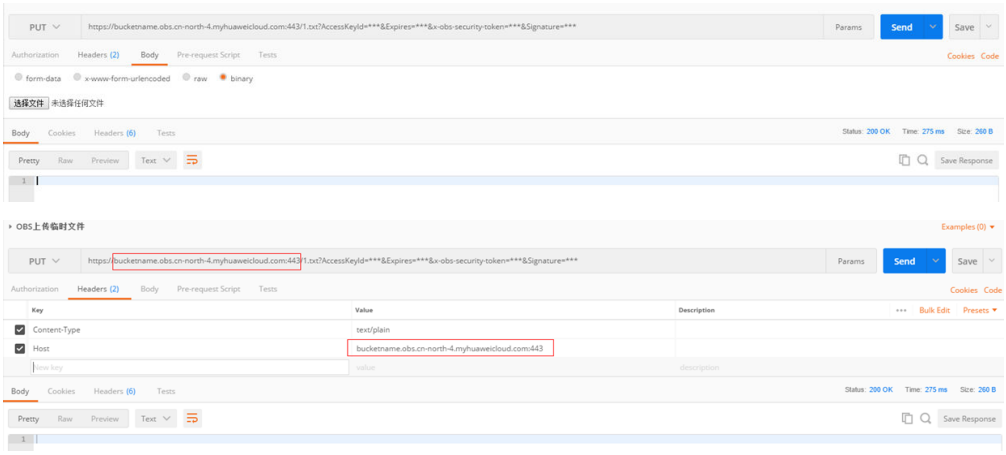
3~4. 设备请求上传文件URL，平台下发文件上传URL，格式请参考**设备上报获取文件上传URL请求**接口文档。

5.调用OBS接口，使用平台下发的URL上传设备文件，URL有效期时间以下发的expire为准，单位为秒，默认是1个小时。

- 方法一：直接使用该URL，以Postman为例。

  使用PUT方法调用URL，body选择binary，选择具体的文件上传，文件名与上报的文件名一样，这样在OBS才可以看到对应的文件。

  调用接口的header可以不带Content-Type或者Host。如果必须要带，请检查Content-Type是否为text/plain，以及Host是否为URL的域名，否则调用接口将返回403状态码SignatureDoesNotMatch错误。



- 方法二：集成OBS的SDK调用接口。

  参考**使用URL进行授权访问**，使用put请求上传对象SDK来上传对象。

6~7. 设备请求下载存储在OBS服务的文件，平台下发文件下载URL，格式请参考**平台下发文件上传临时URL**接口文档。

8. 调用OBS接口，使用平台下发的URL下载设备文件。

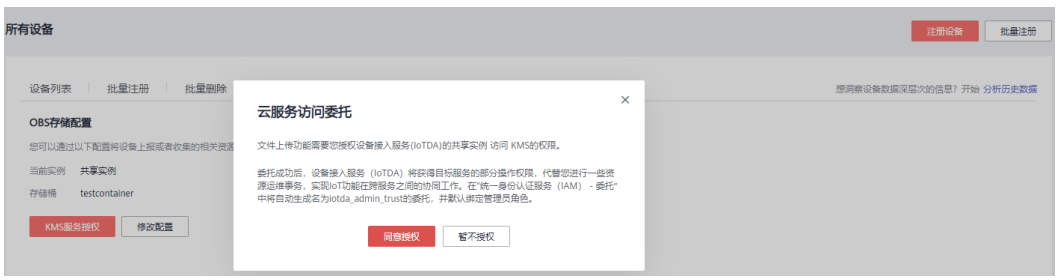
- 方法一：使用GET方法调用URL。调用接口的header可以不带Content-Type或者Host。如果必须要带，请检查Content-Type是否为text/plain，以及Host是否为URL的域名，否则调用接口将返回403状态码SignatureDoesNotMatch错误。
- 方法二：集成OBS的SDK调用接口，使用GET请求下载对象SDK来下载文件对象。

配置文件上传功能

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏，选择“设备 > 所有设备”，单击“文件上传”页签。

步骤3 单击“服务授权”，在新弹出的页面中单击“同意授权”。



注：如用户仅授权过设备接入服务访问对象存储服务的权限，可在控制台选择“设备 > 所有设备”，单击“文件上传”页签，然后单击“KMS服务授权”按钮。授权设备接入服务访问密钥管理服务的权限。

步骤4 （可选）若没有桶，请先在OBS服务控制台创建桶，已有OBS桶的话，请跳过此步骤。

1. 访问**OBS控制台**。
2. 单击右上角的“创建桶”，进入参数配置页面，更新参数说明请参考**创建桶**。

说明

使用OBS云服务管理文件，将由OBS服务进行收费，设备接入服务不再单独对文件存储进行收费。更多OBS文件存储计费详情，请参考**计费说明**。

步骤5 单击“OBS存储配置”，选择正确桶后，该实例下的所有设备文件将上传到配置的OBS桶，您也可以单击“修改配置”更改桶。

说明

调用OBS接口上传设备文件时，每次只能上传一个文件，且文件大小不能超过5GB。

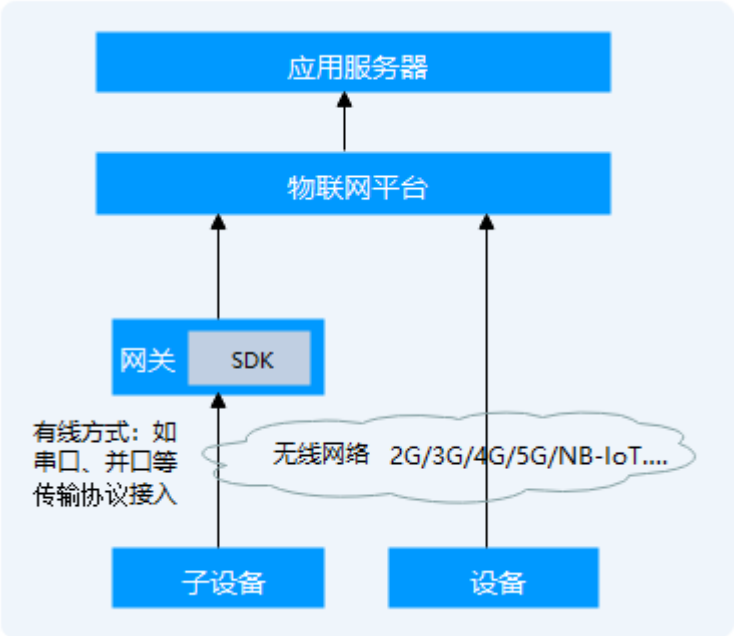
----结束

3.10 网关与子设备

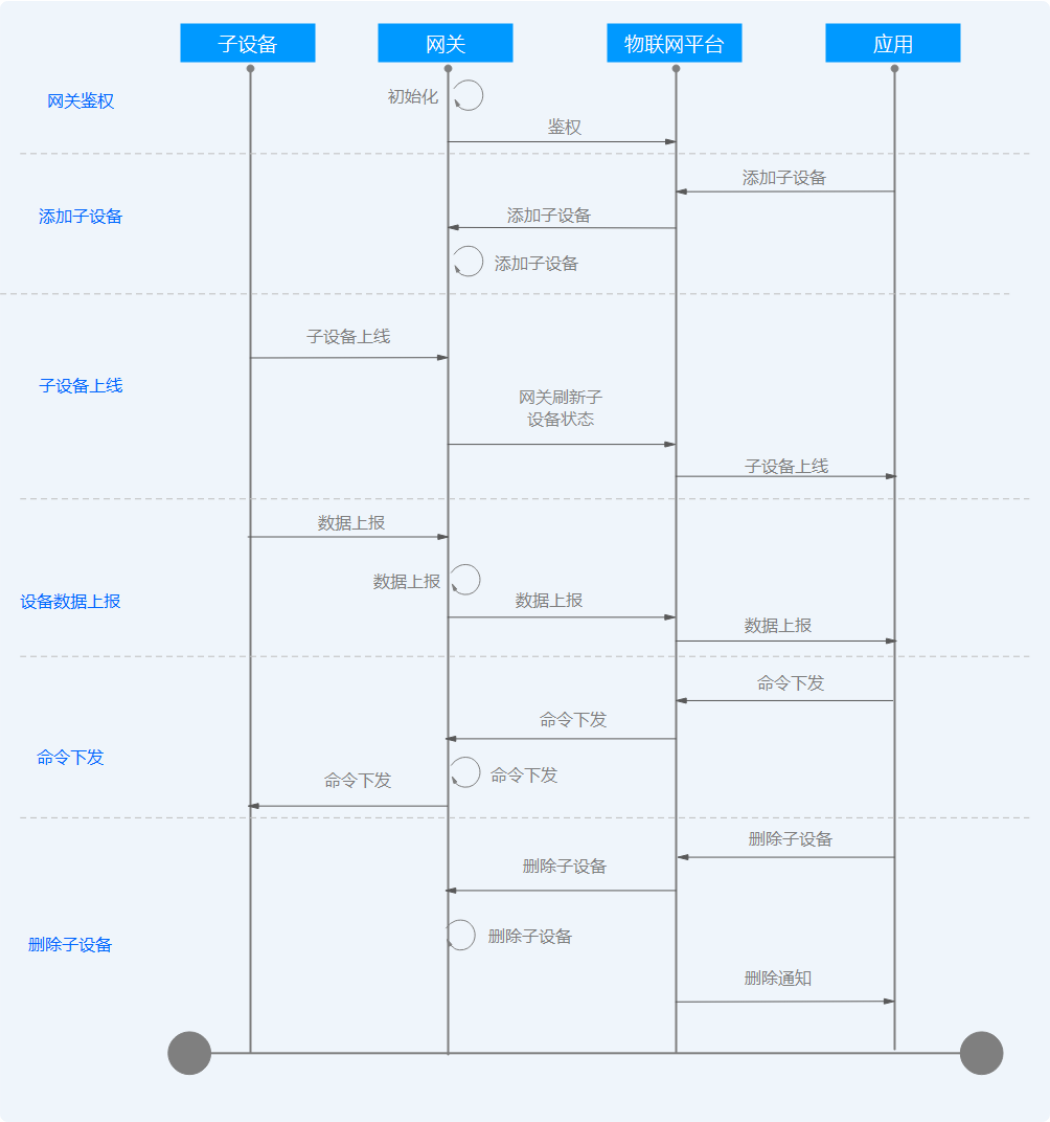
概述

物联网平台支持设备直连，也支持设备挂载在网关上，作为网关的子设备，由网关直连。

针对未实现TCP/IP协议栈的设备，由于无法直接同物联网平台通信，它需要通过网关进行数据转发，如下图所示。



业务流程



IoT Device SDK提供相关的接口，开发者调用这些接口网关实现与物联网平台的对接。不同语言的SDK的接口名称不一样，每个接口功能请参见**IoT Device SDK使用指南（Java）**、**IoT Device SDK使用指南（C）**、**IoT Device SDK使用指南（C#）**、**IoT Device SDK使用指南（Android）**、**IoT Device SDK Tiny使用指南（C）**。

1. 在物联网平台上传网关的产品模型，并注册网关设备。
2. 网关调用鉴权接口上线。
3. 在物联网平台上传子设备的产品模型。
4. 网关鉴权成功后，应用调用**添加子设备**接口，填写设备相关信息（与产品模型定义一致）。添加成功后，您可以在物联网平台查看添加的子设备（**如何查看？**）。您也可以通过控制台添加子设备，详细请查看**如何添加?**
5. 添加子设备后，子设备状态显示“未激活”。请在子设备添加成功后，或者子设备上报数据前，调用**网关更新子设备状态**接口进行设备状态更新。

说明

子设备的状态表示子设备接入网关的状态，由网关上报到物联网平台进行状态的刷新；如果网关不能正常上报子设备的状态信息到物联网平台，则展示的子设备状态不会刷新。例如：某子设备通过网关接入到物联网平台，子设备状态为在线状态，如果此时网关与物联网平台断开连接，则网关不能上报子设备的状态到物联网平台，该子设备的状态会一直显示在线。

6. 网关调用**批量属性上报**接口上报子设备的数据，接口里的参数填写网关和子设备的相关设备信息。
7. 网关订阅命令下发Topic，接收并处理应用服务器或物联网平台下发的命令。
8. 应用服务器调用**删除设备**接口，给网关下发删除子设备命令，网关收到该命令后，可以进行相应的业务处理。

网关如何接入物联网平台?

通过在网关上集成SDK, 设备将数据上报给网关，通过网关转发到华为物联网平台，网关接入物联网平台的开发指南请参见**设备通过网关接入平台**。

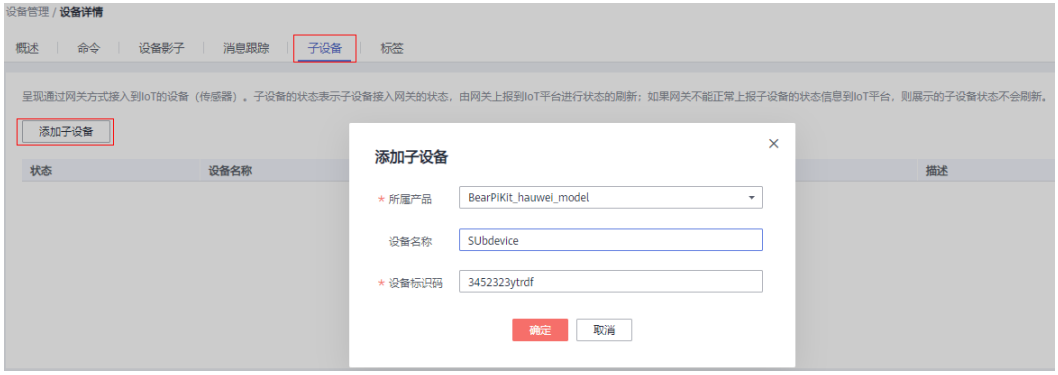
物联网平台上如何添加子设备?

方式1

当网关接入物联网平台后，调用**创建设备**接口，完成子设备接入到物联网平台。

方式2

访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台，选择“设备 > 所有设备”，在设备列表中，单击具体的网关设备进入到网关的详情页面，进入“子设备”页签，单击“添加子设备”。



查看子设备

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择“设备 > 所有设备”页面，在设备列表中，单击具体的网关进入到网关的详情页面，选择“子设备”页签。

步骤3 “子设备”页签呈现通过该网关方式接入到物联网平台的设备，可以查看子设备的状态、设备ID、设备类型等信息。

步骤4 在子设备页签中，单击具体的子设备，可以查看子设备的**设备详情**。

----结束

4 消息通信

4.1 概述

物联网平台具有双向设备通信的功能。一方面您可以通过设备侧接口，将数据上报到物联网平台，再通过订阅推送把数据推送到您的应用服务器，或者转发到其他华为云通道。另一方面，您可以通过应用侧接口或控制台，将命令下发给设备，实现对设备远程控制。

消息通信是基于物模型设计的，数据上行主要分为属性上报、消息上报，数据下行主要分为命令下发、消息下发、属性修改（下发）。物模型包含属性、命令，事件，超出物模型定义之外的包含消息。

表 4-1 消息通信说明

数据类型	消息类型	差异点	相同点
数据上行	属性上报	依赖产品模型，需要与产品模型中定义的属性相匹配，支持在控制台的设备详情中查看最新上报的快照数据，支持推送给已订阅的应用侧，支持通过数据分析服务查看历史数据。	都可以通过设备侧接口上报数据给平台，都支持规则转发。
	消息上报	不依赖产品模型，平台对消息内容不进行校验，不支持在控制台的设备详情中查看最新上报的快照数据，不支持通过数据分析服务查看历史数据。	
数据下行	命令下发	依赖产品模型，需要与产品模型中定义的的命令相匹配，是同步的（命令下发后等待设备侧的回复响应，若无响应则超时失败），支持控制台下发命令。	都可以通过应用侧接口调用给设备下发指令。
	属性修改	依赖产品模型，需要与产品模型中定义的属性相匹配，是同步的（属性修改后等待设备侧的回复响应，若无响应则超时失败），支持控制台属性配置（设备影子），支持通过设备影子修改设备属性。	

数据类型	消息类型	差异点	相同点
	消息下发	不依赖产品模型，平台下发消息给设备，是异步的（消息下发后无需等待设备侧回复响应），不支持控制台下发消息。	

4.2 自定义 Topic 通信

概述

使用MQTT协议接入的设备，平台和设备之间基于Topic进行通信。Topic分为自定义Topic和系统Topic。系统Topic为平台预置的基本通信Topic，您也可以根据实际业务需要，在平台自定义Topic

Topic 分类

Topic类别	描述
系统Topic	平台预先定义了各种设备和平台通信的Topic，具体Topic列表和功能说明可参考**Topic定义**。
自定义Topic	用户可以自定义Topic，设备和平台间可以基于用户自定义的Topic进行通信。 自定义Topic适用于： • 基于自定义Topic进行消息流转 • 通过自定义Topic进行消息下发

使用限制

每个产品模型最多支持10个自定义Topic。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择MQTT协议类产品，在产品详情页中，选择“Topic 管理”“自定义Topic”。

步骤3 在弹出的页面中，选择设备操作权限，填写Topic名称。

参数名称	描述
Topic名称	Topic的前缀已经规定好，固定为：$oc/devices/{device_id}/user/，其中{device_id}为标识符变量，实际发布和订阅过程中需要替换为实际的设备ID。用户自定义Topic的格式必须以“/”进行分层。 长度限制为1-64位，只允许输入数字,大小写字母,下划线,斜杠符，斜杠符不能连续。
设备操作权限	• 发布：设备侧消息上报时，可按配置中自定义的Topic进行消息上报；数据流转时，设备消息中会携带Topic参数标识该消息从哪个Topic上报。 • 订阅：设备侧消息下发时，可在消息内容中指定此消息下发使用哪个Topic；消息发往设备时，可以根据指定的Topic下发。 • 发布和订阅：同时具备发布和订阅的权限。
描述	关于该Topic的描述。

步骤4 单击“确定”，完成新增自定义Topic。

自定义Topic添加成功后，您可以在自定义Topic列表执行修改和删除操作。

----结束

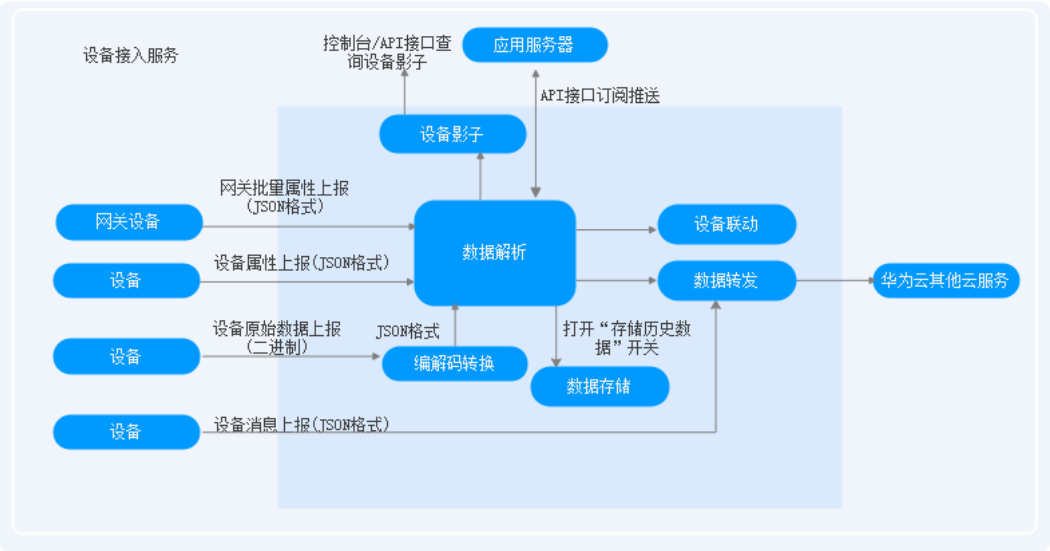
查看**使用自定义Topic进行通信**的最佳实践，了解自定义Topic的使用方法。如果无法使用前缀为$oc/devices/{device_id}/user/的TOPIC，需要完全自定义TOPIC内容，可以参考链接**设备自定义TOPIC迁移上云**。

4.3 数据上报

概述

当设备和物联网平台完成对接后，设备可通过以下方式发送数据到物联网平台：

- 设备消息上报：设备可以通过消息上报接口将设备的自定义数据上报到平台，平台对设备上报的消息不进行解析和存储，通过**数据转发规则**转发到华为云其他云服务上进行存储和处理，然后通过其他云服务的控制台或者API接口进行进一步的数据处理。
- 设备原始数据（二进制）上报：设备可以通过二进制上报接口上报设备的原始码流，平台通过编解码插件将设备原始数据解析为产品模型定义的JSON格式后，再进行后续处理。
- 设备属性上报：设备通过属性上报接口，将产品模型中定义的属性数据上报给平台，平台解析后再进行后续处理。
- 网关批量属性上报：网关设备将批量设备的数据一次性上报到平台，平台解析后再进行后续处理。



说明

CoAP协议接入的设备考虑到功耗和带宽的影响，当前只支持设备上报原始的二进制码流，由平台根据编解码插件，将码流转换为产品模型中定义的JSON格式数据后，再进行后续处理。

查看设备影子

如果已开通物联网数据分析服务，则可以针对此设备最近一段时间上报的历史数据进行时序洞察分析。

说明

查看历史数据是自动跳转到IoT数据分析服务，支持属性上报的数据和消息上报的数据。

若要正常使用数据分析服务查看针对属性和消息的分析，需要提前完成：

- 已完成产品模型开发。
- 属性上报时，上报的属性跟产品模型中定义的一致。
- 对于消息上报的数据以及无产品模型的属性上报的数据，IoT数据分析服务默认以数据的原始格式保存，请在IoT数据分析服务中定义需要进行数据分析的字段格式。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在设备列表中，单击具体的设备进入到设备的详情页面，单击“查看历史数据”跳转到数据分析服务进行时序洞察分析。

另外，界面上还支持查看当前资源空间下所有设备的历史数据，并支持导出历史数据。

----结束

相关应用侧 API 接口

- **查询设备属性**
- **查询设备消息**
- **查询设备**
- **查询设备影子数据**

相关设备侧 MQTT 接口

- **设备消息上报**
- **设备属性上报**
- **网关批量设备属性上报**

4.4 编解码插件

什么是编解码插件

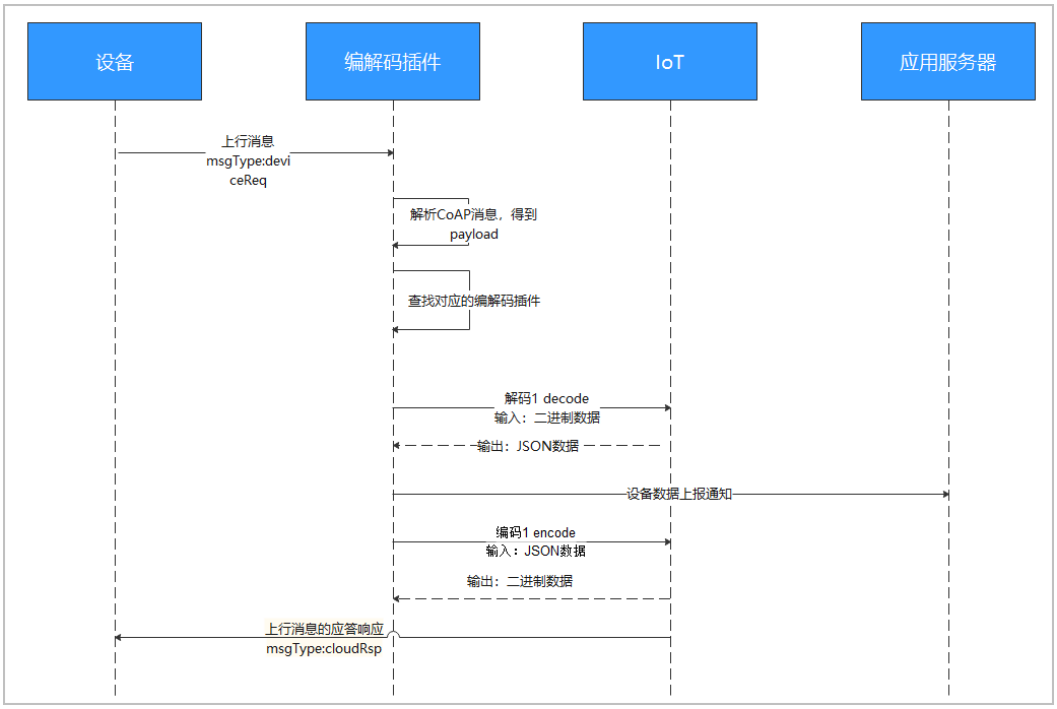
一款产品的设备上报数据时，如果“数据格式”为“二进制码流”，则该产品需要进行编解码插件开发；如果“数据格式”为“JSON”，则该产品不需要进行编解码插件开发。

以NB-IoT场景为例，NB-IoT设备和物联网平台之间采用CoAP协议通讯，CoAP消息的payload为应用层数据，应用层数据的格式由设备自行定义。由于NB-IoT设备一般对

省电要求较高，所以应用层数据一般不采用流行的JSON格式，而是采用二进制格式。但是，物联网平台与应用侧使用JSON格式进行通信。因此，您需要开发编码插件，供物联网平台调用，以完成二进制格式和JSON格式的转换。

数据上报流程

图 4-1 数据上报流程图

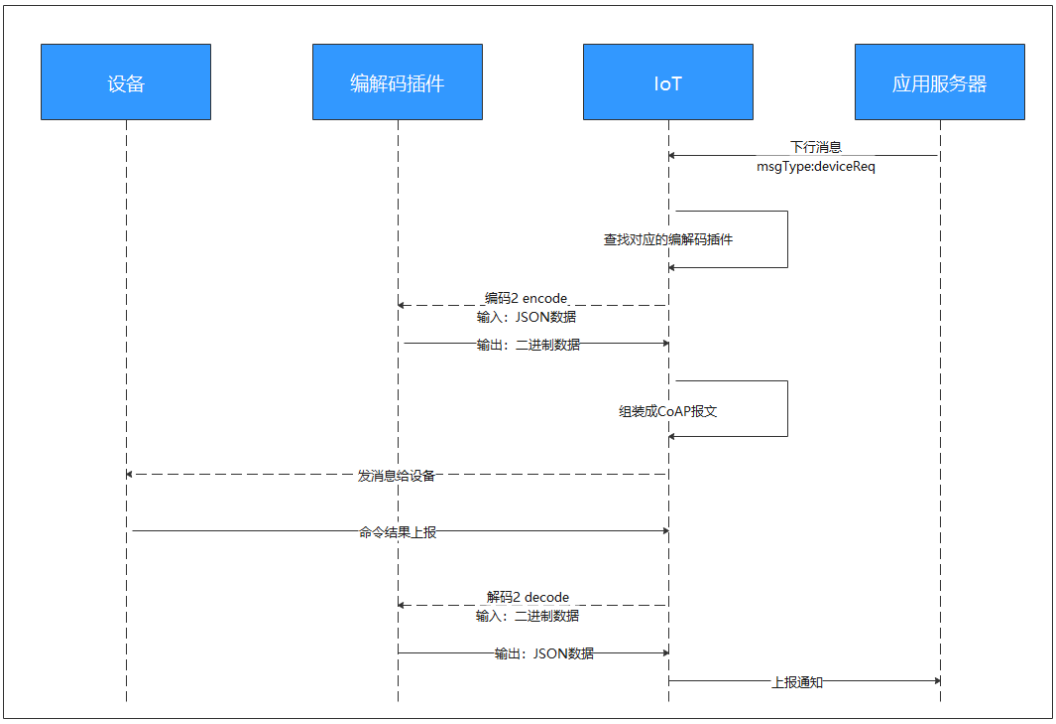


在数据上报流程中，有两处需要用到编解码插件：

- 将设备上报的二进制码流解码成JSON格式的数据，发送给应用服务器。
- 将应用服务器响应的JSON格式数据编码成二进制码流，下发给设备。

命令下发流程

图 4-2 命令下发流程图



在命令下发流程中，有两处需要用到编解码插件：

- 将应用服务器下发的JSON格式数据编码成二进制码流，下发给设备。
- 将设备响应的二进制码流解码成JSON格式的数据，上报给应用服务器。

编解码插件开发方法

物联网平台提供了多种开发编解码插件的方法，您可以根据自己需求，选择对应的方法开发编解码插件。由于离线开发编解码插件的方法较为复杂，且耗时比较长，我们**推荐**使用**图形化开发编解码插件**和**脚本化开发**。

- **图形化开发**：是指在设备接入控制台，通过可视化的方式快速开发一款产品的编解码插件。详细请参考**图形化开发**。
- **离线开发**：是指使用编解码插件的Java代码Demo进行二次开发，实现编解码功能、完成插件打包和质检等。详细请参考**离线开发**。
- **脚本化开发**：是指使用JavaScript脚本实现编解码的功能。详细请参考**脚本化开发**。
- FunctionGraph开发：是指通过FunctionGraph来实现编解码的功能。详细请参考FunctionGraph开发。

4.5 命令下发

4.5.1 命令下发机制

为能有效地对设备进行管理，设备的产品模型中定义了物联网平台可向设备下发的命令，应用服务器可以调用物联网平台开放的API接口向设备下发命令，以实现对设备的远程控制。

物联网平台有同步命令下发和异步命令下发两种命令下发机制，如下表所示。

命令下发机制	定义	适用场景	LwM2M/CoAP协议设备	MQTT协议设备
同步命令下发	应用服务器可调用同步命令下发接口向指定设备下发命令，以实现对设备的同步控制。平台负责将命令以同步方式发送给设备，并将设备执行命令结果在HTTP请求中同步返回, 如果设备没有响应，平台会返回给应用服务器超时。	同步命令下发适合对命令实时性有要求的场景，比如路灯开关灯，燃气表开关阀。使用同步命令下发时，命令下发的时机需要由应用服务器来保证。	不适用	适用
异步命令下发	应用服务器可调用异步命令下发接口向指定设备下发命令，以实现对设备的控制。平台负责将命令发送给设备，并将命令执行结果异步推送给应用。 异步命令下发又分为缓存下发和立即下发。 ● 立即下发：不管设备是否在线，平台收到命令后立即下发给设备。如果设备不在线或者设备没收到指令则下发失败。 ● 缓存下发：物联网平台在收到命令后先缓存，等设备上线或者设备上报属性时再下发给设备，如果单个设备存在多条缓存命令，则进行排队串行下发。	● 立即命令下发适用于实时性要求高的场景； ● 缓存下发适合对命令实时性要求不高的场景，比如配置水表的参数。	适用	不适用

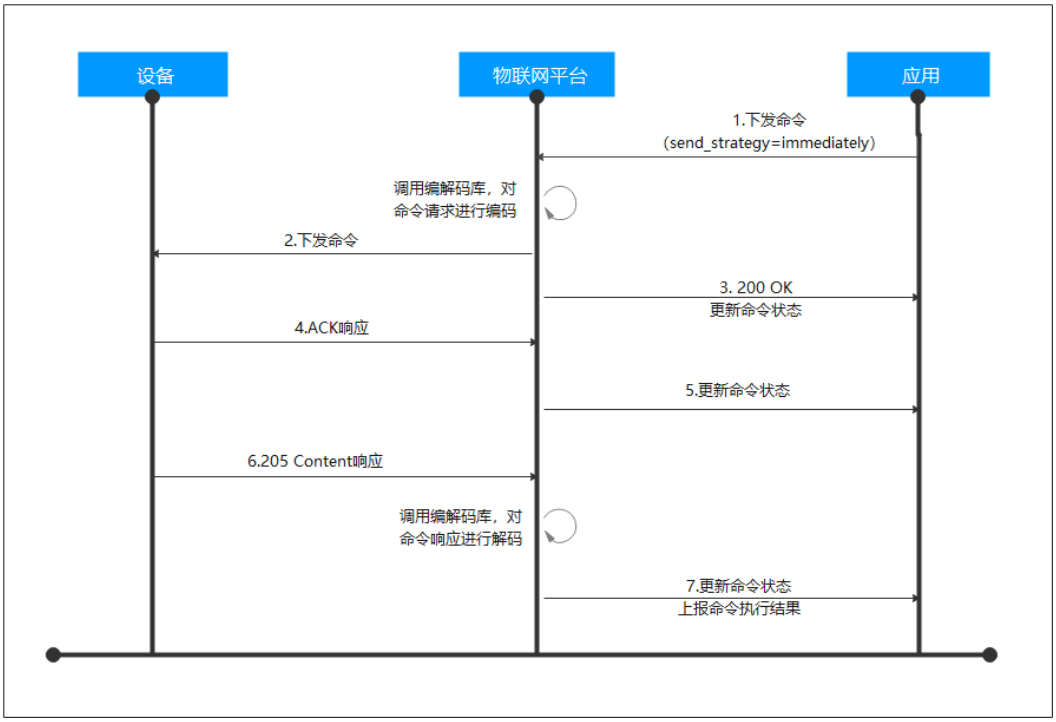
4.5.2 LwM2M/CoAP 设备命令下发

针对LwM2M/CoAP协议的设备，物联网平台提供立即下发和缓存下发两种机制。应用服务器向物联网平台下发命令时，携带参数send_strategy（表示下发策略）。

- send_strategy=immediately：命令立即下发。
- send_strategy=delay：命令缓存下发。

LwM2M/CoAP 设备命令立即下发

本文介绍命令立即下发流程。



1. 应用调用**下发异步设备命令**接口，下发请求到物联网平台，携带send_strategy为immediately。消息样例如下：

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/async-commands
Content-Type: application/json
X-Auth-Token: ********

{
  "service_id" : "WaterMeter",
  "command_name" : "ON_OFF",
  "paras" : {
    "value" : "ON"
  },
   "expire_time": 0,
   "send_strategy": immediately
}
```

2. 物联网平台调用**编解码插件**对命令请求进行编码后，会通过LwM2M协议定义的**设备管理和服务实现**接口的**Execute**操作下发命令，消息体为二进制格式。
3. 物联网平台向应用返回200 OK，携带命令状态为**SENT**。（如果设备不在线或者设备没收到指令则下发失败，命令状态为**FAILED**）
4. 设备收到命令后返回ACK响应。
5. 若应用订阅了命令的状态变更通知，物联网平台通过命令状态更新通知接口推送消息给应用，携带命令状态为**DELIVERED**。消息样例如下：

```
Method: POST
request:
Body:
{
   "resource": "device.commmad.status",
```

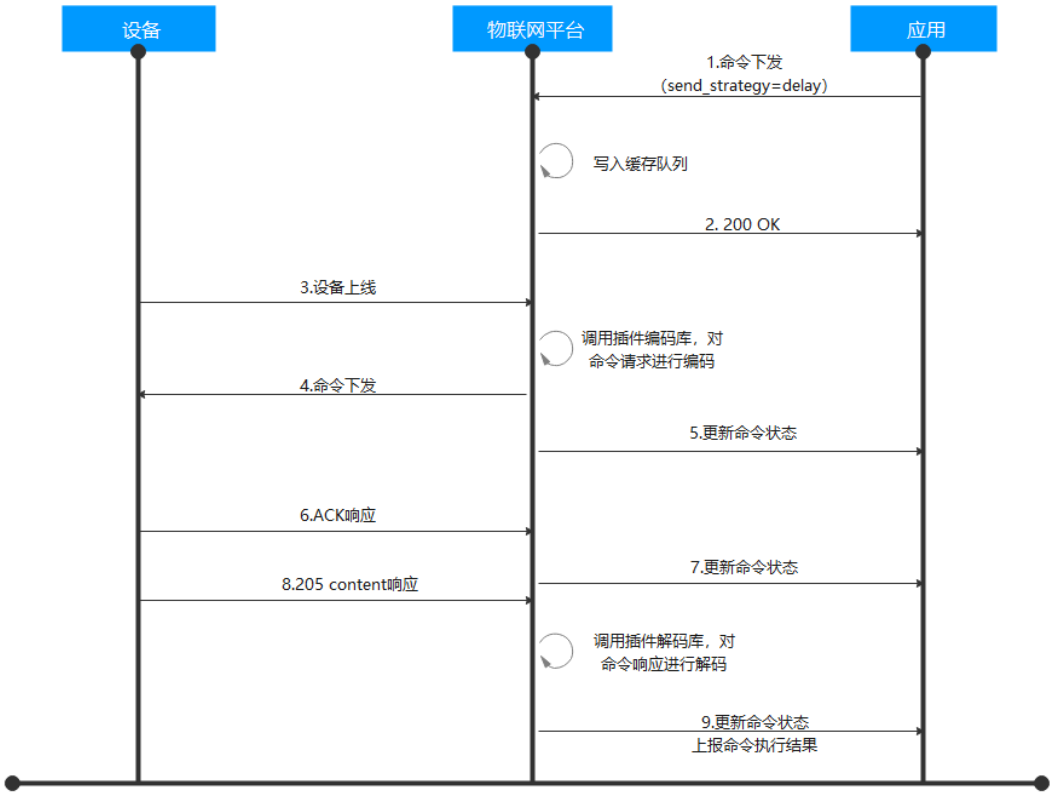
```
    "event": "update",
    "event_time": "20200811T080745Z",
    "notify_data": {
        "header": {
            "app_id": "8d4a34e5363a49bfa809c6bd788e6ffa",
            "device_id": "5f111a5a29c62ac7edc88828_test0001",
            "node_id": "test0001",
            "product_id": "5f111a5a29c62ac7edc88828",
            "gateway_id": "5f111a5a29c62ac7edc88828_test0001",
            "tags": []
        },
        "body": {
            "command_id": "49ca40af-7e14-4f7b-b97b-78cdd347a6b9",
            "created_time": "20200811T080738Z",
            "sent_time": "20200811T080738Z",
            "delivered_time": "20200811T080745Z",
            "response_time": "",
            "status": "DELIVERED",
            "result": null
        }
    }
}
```


6. 设备执行命令后通过205 Content响应返回命令执行结果。
7. 若应用订阅了命令的状态变更通知，物联网平台会调用编解码插件对设备响应进行解码，然后通过命令状态更新通知接口推送消息给应用，携带命令状态为**SUCCESSFUL**。消息样例如下：


```
Method: POST
request:
Body:
{
    "resource": "device.commmad.status",
    "event": "update",
    "event_time": "20200811T080745Z",
    "notify_data": {
        "header": {
            "app_id": "8d4a34e5363a49bfa809c6bd788e6ffa",
            "device_id": "5f111a5a29c62ac7edc88828_test0001",
            "node_id": "test0001",
            "product_id": "5f111a5a29c62ac7edc88828",
            "gateway_id": "5f111a5a29c62ac7edc88828_test0001",
            "tags": []
        },
        "body": {
            "command_id": "49ca40af-7e14-4f7b-b97b-78cdd347a6b9",
            "created_time": "20200811T080738Z",
            "sent_time": "20200811T080738Z",
            "delivered_time": "20200811T080745Z",
            "response_time": "20200811T081745Z",
            "status": "SUCCESSFUL",
            "result": {
                "resultCode":"SUCCESSFUL",
                "resultDetail": {
                    "value": "ON"
                }
            }
        }
    }
}
```


LwM2M/CoAP 设备命令缓存下发

本文介绍缓存命令下发流程。



1. 应用调用**下发异步设备命令**接口，下发请求到物联网平台，携带send_strategy为delay。
2. 物联网平台将命令写入缓存队列，并上报200 OK，携带命令状态为**PENDING**。
3. 设备上线或设备上报数据到平台。
4. 物联网平台调用**编解码插件**对命令请求进行编码后，根据协议规范下发命令给设备。
5. 若应用订阅了命令的状态变更通知，物联网平台通过命令状态变化通知接口推送消息给应用，携带命令状态为**SENT**。
6. 后续流程请参考“命令立即下发”的步骤**4**到步骤**7**。

下发单个 LwM2M/CoAP 设备异步命令下发

平台支持通过调用**下发异步设备命令**接口和在控制台上创建命令下发任务，创建单个LwM2M/CoAP设备的命令下发任务。本文介绍如何在控制台上创建命令下发业务。

方式一：

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

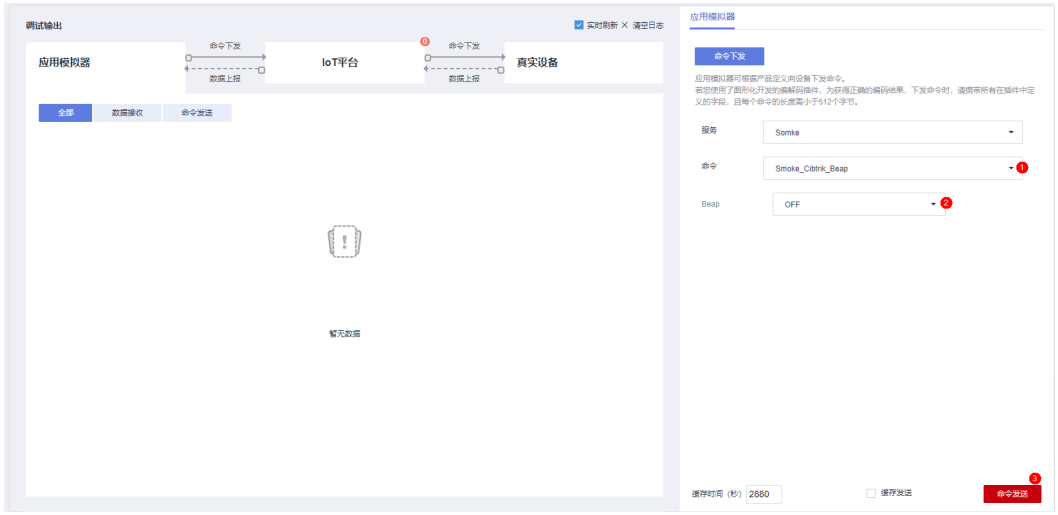
步骤2 在左侧导航栏选择“产品”，单击已创建的产品进入产品详情页面。创建产品操作指导，请参见**创建产品**。

步骤3 在“在线调试”页签中，单击“新增测试设备”，在弹出的窗口中输入LwM2M/CoAP设备参数，然后单击“确认”。

参数	描述
设备类型	选择“真实设备”
设备名称	自定义，如：TestDevice
设备标识码	LwM2M/CoAP设备的IMEI号，请在设备上获取。
设备注册方式	选择“不加密”（为了方便演示当前选择不加密方式，商用场景建议选择加密方式）。

步骤4 在测试列表页面，单击“调试”进入调试页面。

步骤5 选择需要下发的命令并设置命令参数，您可以选择立即发送或者缓存发送。



步骤6 通过在控制台查看下发的历史命令列表，通过该功能可以详细查看命令下发任务的创建时间、平台发送命令的时间、送达的时间、发送的状态等信息，便于用户了解命令的**执行状态**。

----结束

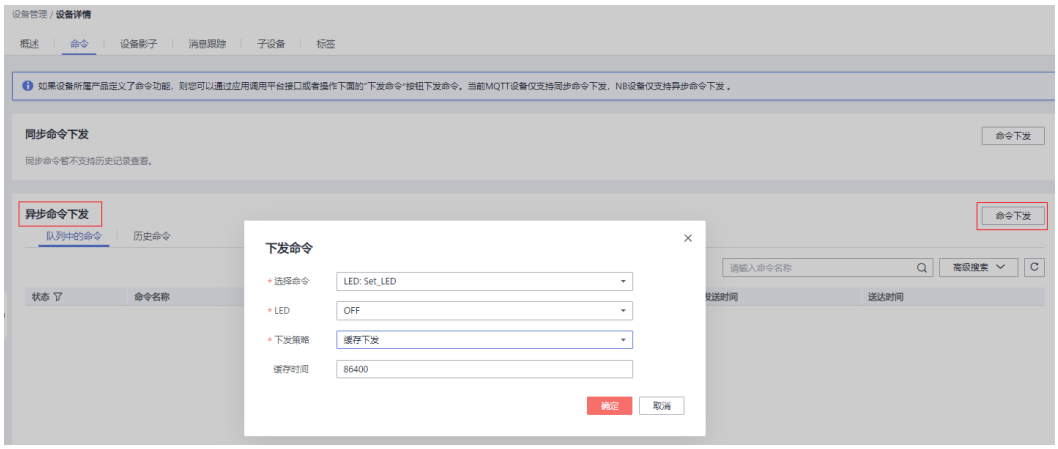
同时，还支持通过调用**查询设备命令**接口，在物联网平台查询下发命令的状态及内容信息，以了解命令的执行情况。

方式二：

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 单击“设备”，在设备列表中，单击具体的设备进入到设备的详情页面。

步骤3 在“命令”页签，单击异步命令下发右侧的“命令下发”，在弹出的窗口中选择需要下发的命令并设置命令参数。您可以选择立即下发或者缓存下发。



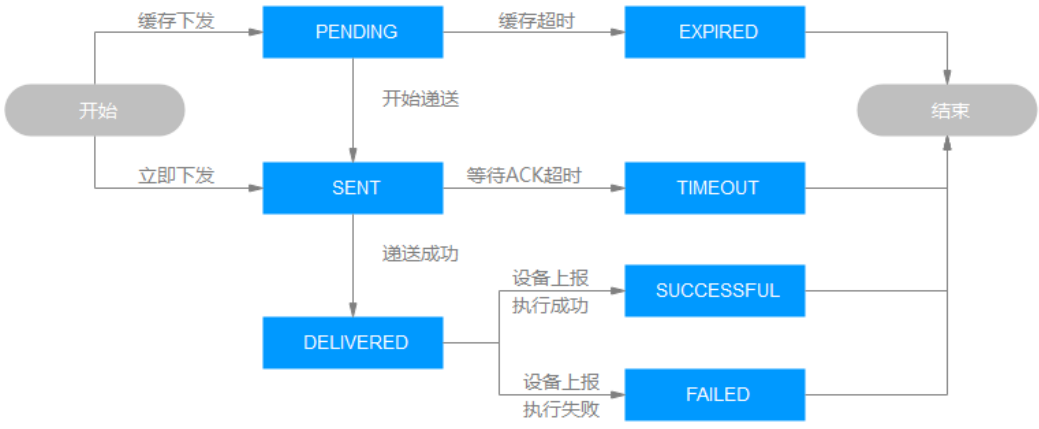
说明

- 设备支持的命令，需要由**产品模型**定义后，才能在此进行命令下发操作。
- LwM2M/CoAP设备仅支持异步命令下发 ，MQTT设备的同步命令下发请前往**MQTT设备下发命令**。

----结束

LwM2M/CoAP 协议设备命令执行状态说明

命令执行状态以及状态变化机制如下所示。



命令执行状态	说明
等待（PENDING）	• LwM2M/CoAP设备采用缓存下发模式下发命令时，如果设备未上报数据，物联网平台会将命令进行缓存，此时任务状态为“等待”状态。 • LwM2M/CoAP设备采用立即下发模式下发命令时，无此状态。
超期（EXPIRED）	• LwM2M/CoAP设备采用缓存下发模式下发命令时，如果在设置的超期时间内，物联网平台未将命令下发给设备，则状态变更为“超期”。超期时间会根据应用侧接口中携带的expireTime为准，如果未携带，默认24h。 • LwM2M/CoAP设备采用立即下发模式下发命令时，无此状态。
已发送（SENT）	• LwM2M/CoAP设备采用缓存下发模式下发命令时，设备上报数据，物联网平台会将缓存的命令发送给设备，此时状态会由“等待”变为“已发送”。 • LWM2M/CoAP设备采用立即下发模式下发命令时，如果设备在线，状态为“已发送”。
超时（TIMEOUT）	LwM2M/CoAP设备收到命令后，物联网平台在180秒内未收到设备反馈的收到命令响应，此时状态会变为“超时”。
已送达（DELIVERED）	物联网平台收到设备反馈的已收到下发命令响应后，状态变为“已送达”。
成功（SUCCESSFUL）	如果设备在执行完命令后，会给物联网平台反馈命令执行成功的结果，将任务状态变更为“成功”。
失败（FAILED）	• 如果设备在执行完命令后，会给物联网平台反馈命令执行失败的结果，将任务状态变更为“失败”。 • LwM2M/CoAP设备采用立即下发模式下发命令时，如果设备离线，状态为“失败”。

相关 API 接口

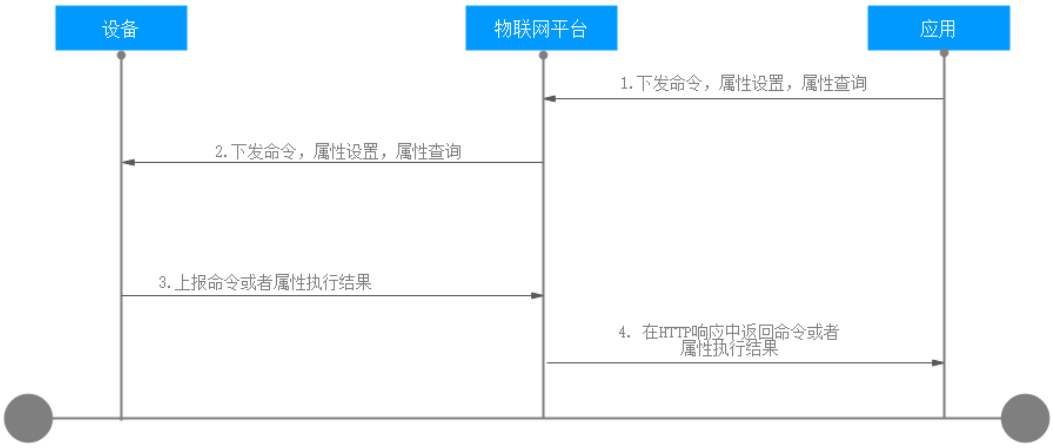
- 下发异步设备命令
- 查询命令详情

4.5.3 MQTT 设备命令/属性/消息下发

物联网平台支持MQTT协议设备进行命令下发，属性设置，属性查询以及消息下发等操作。

MQTT 设备命令下发/属性设置/属性查询

命令下发，属性设置，属性查询业务流程相同，应用下发命令之后， 应用在请求中同步等待设备的命令执行结果。 本文仅以下发命令为例，属性设置和属性查询分别参考修改设备属性和查询设备属性接口。



1. 应用调用**下发设备命令**接口，下发请求到物联网平台，命令下发消息样例如下：

```
POST https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}/commands
Content-Type: application/json
X-Auth-Token: ********

{
  "service_id" : "WaterMeter",
  "command_name" : "ON_OFF",
  "paras" : {
    "value" : "ON"
  }
}
```

2. 物联网平台根据协议规范下发命令给设备。

   MQTT设备必须先订阅**平台命令下发**下行接口对应的Topic才能收到平台下发的命令，消息样例如下：

```
Topic: $oc/devices/{device_id}/sys/commands/request_id={request_id}
数据格式：
{
  "object_device_id": "{object_device_id}",
  "command_name": "ON_OFF",
  "service_id": "WaterMeter",
  "paras": {
    "value": "ON"
  }
}
```

3. 设备执行命令后通过**平台命令下发**上行接口返回命令执行结果，消息样例如下：

```
Topic：  $oc/devices/{device_id}/sys/commands/response/request_id={request_id}
数据格式：
{
    "result_code": 0,
    "response_name": "COMMAND_RESPONSE",
    "paras": {
        "result": "success"
    }
}
```

4. 应用侧收到发送HTTP下发命令的同步响应结果。消息样例如下：

```
Status Code: 200 OK
Content-Type: application/json

{
  "command_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "response" : {
    "result_code" : 0,
    "response_name" : "COMMAND_RESPONSE",
    "paras" : {
      "result" : "success"
    }
```

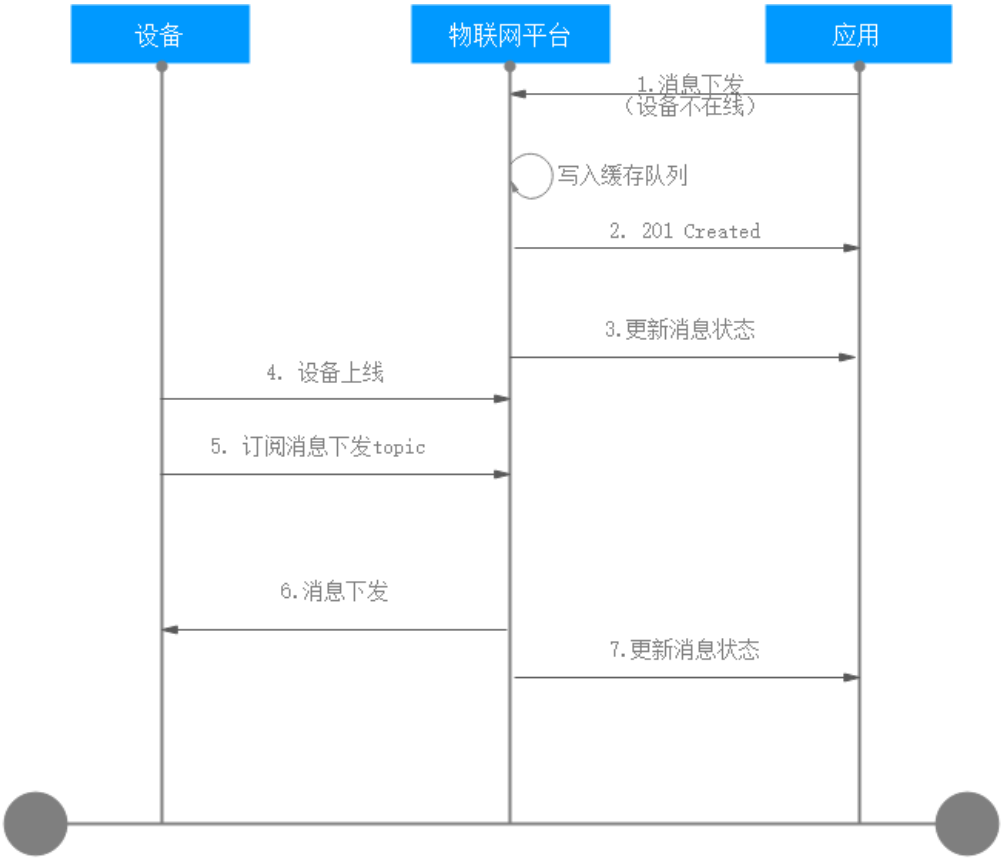
```
  }
}
```

MQTT 设备消息下发

MQTT设备消息下发分为缓存下发和立即下发， 当设备在线时会立即下发， 当设备不在线时会先把消息缓存起来，等设备上线后下发。

- **MQTT协议设备消息缓存下发**

  本文介绍MQTT协议缓存消息下发流程。



a. 应用调用**下发设备消息**接口，下发请求到物联网平台，下发消息样例如下：

```
POST https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}/messages
Content-Type: application/json
X-Auth-Token: ********

{
  "message_id": "99b32da9-cd17-4cdf-a286-f6e849cbc364",
  "name": "messageName",
  "message": "HelloWorld",
  "topic": "messageDown"
}
```

b. 物联网平台向应用返回201 Created，消息状态为PENDING。

c. 物联网平台推送消息结果给应用，消息样例如下：

```
Topic: $oc/devices/{device_id}/sys/messages/down
数据格式：
{
    "resource": "device.message.status",
    "event": "update",
    "notify_data": {
      "message_id": "string",
      "name": "string",
```

```
    "device_id": "string",
    "status": "PENDING",
    "timestamp": "string"
   }
}
```

d. 设备上线。

e. 设备订阅消息下发的topic，用于接收消息，订阅的topic见步骤**6**。

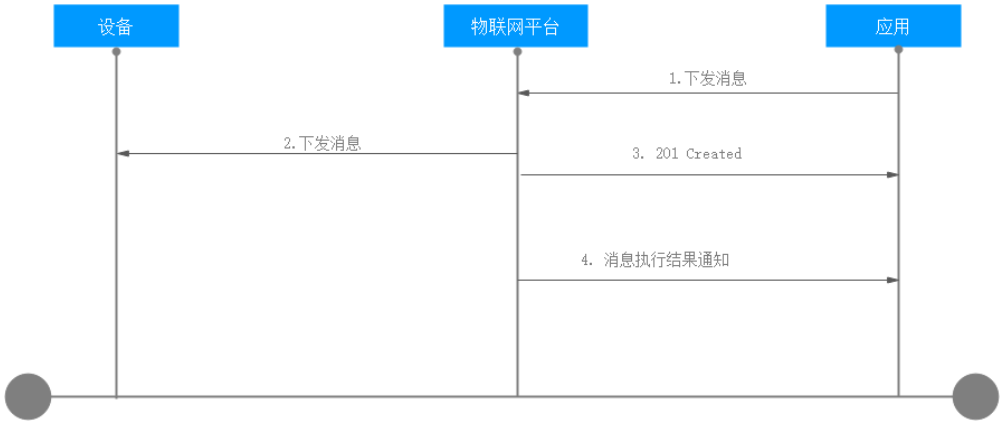
f. 物联网平台根据协议规范下发消息给设备。MQTT设备必须先订阅**平台消息下发**下行接口对应的Topic才能收到平台下发的消息，消息样例如下：

```
Topic: $oc/devices/{device_id}/sys/messages/down
数据格式：
{
   "object_device_id": "{object_device_id}",
   "name": "name",
   "id": "id",
   "content": "hello"
}
```

7. 平台将消息的最终结果推送给应用服务器，消息结构参考步骤**3**。

- **MQTT协议设备消息立即下发**

  本文介绍MQTT协议消息立即下发流程。



a. 应用调用**下发设备消息**接口，下发请求到物联网平台，下发消息样例如下：

```
POST https://{Endpoint}/v5/iot/{project_id}/devices/{device_id}/messages
Content-Type: application/json
X-Auth-Token: ********

{
  "message_id": "99b32da9-cd17-4cdf-a286-f6e849cbc364",
  "name": "messageName",
  "message": "HelloWorld",
  "topic": "messageDown"
}
```

b. 物联网平台根据协议规范下发消息给设备。MQTT协议设备必须先订阅**平台消息下发**下行接口对应的Topic才能收到平台下发的消息，消息样例如下：

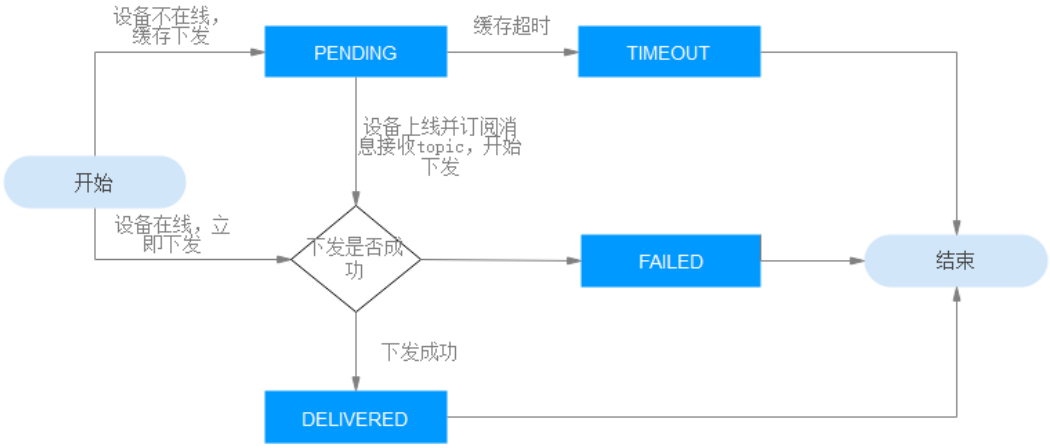
```
Topic: $oc/devices/{device_id}/sys/messages/down
数据格式：
{
   "object_device_id": "{object_device_id}",
   "name": "name",
   "id": "id",
   "content": "hello"
}
```

c. 物联网平台推送消息结果给应用，消息样例如下：

```
Topic: $oc/devices/{device_id}/sys/messages/down
数据格式：
{
```

```
  "resource": "device.message.status",
  "event": "update",
  "notify_data": {
    "message_id": "string",
    "name": "string",
    "device_id": "string",
    "status": "string",
    "timestamp": "string"
   }
}
```

MQTT 设备下发消息状态



消息执行状态	说明
等待（PENDING）	MQTT协议设备不在线，物联网平台会将消息进行缓存，此时任务状态为“等待”状态。
超时（TIMEOUT）	物联网平台缓存的PENDING状态的消息，如果1天之内还没有下发下去，物联网平台会将消息状态设置为"超时”。
已送达（DELIVERED）	物联网平台将消息发送给设备后，状态变为“已送达”。
失败（FAILED）	物联网平台发送消息给设备不成功，消息状态变为“失败”。

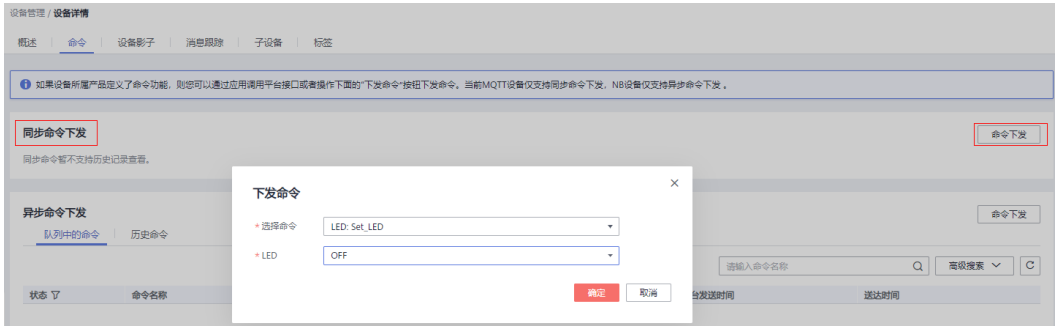
单个 MQTT 设备同步命令下发

平台支持通过调用**下发设备命令**接口和在控制台上创建命令下发任务方式创建单个MQTT协议设备的命令下发功能。本文介绍如何在控制台上创建命令下发业务。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 单击“设备”，在设备列表中，单击具体的设备进入到设备的详情页面。

步骤3 在“命令”页签，单击同步命令下发右侧的“命令下发”，在弹出的窗口中选择需要下发的命令并设置命令参数。



说明

- 设备支持的命令，需要由**产品模型**定义后，才能在此进行命令下发操作。
- MQTT设备仅支持同步命令下发，LwM2M/CoAP设备的异步命令下发，请前往**LWM2M/CoAP设备命令下发**。

----结束

当前不支持通过API接口进行历史命令下发任务的查询，支持历史消息的查询，相关API如下表所示。

API接口	功能
下发设备命令	设备的产品模型中定义了物联网平台可向设备下发的命令，应用服务器可调用此接口向指定设备下发命令，以实现对设备的同步控制。平台负责将命令以同步方式发送给设备，并将设备执行命令结果同步返回, 如果设备没有响应，平台会返回给应用服务器超时。
查询设备属性	设备的产品模型中定义了物联网平台可向设备下发的属性，应用服务器可调用此接口查询指定设备下属性。
修改设备属性	设备的产品模型中定义了物联网平台可向设备下发的属性，应用服务器可调用此接口向指定设备下属性。平台负责将属性以同步方式发送给设备，并将设备执行属性结果同步返回。
查询设备消息	物联网平台可查询指定设备下的消息，平台为每个设备默认最多保存20条消息，超过20条后， 后续的消息会替换下发最早的消息。
下发设备消息	物联网平台可向设备下发消息，应用服务器可调用此接口向指定设备下发消息，以实现对设备的控制。应用将消息下发给平台后，平台返回应用响应结果，平台再将消息发送给设备。
查询指定消息id的消息	物联网平台可查询指定消息id的消息。

批量 MQTT 设备同步命令下发

平台支持通过调用**创建批量任务**接口，对多个MQTT协议设备下发同步命令。本文介绍如何调用**创建批量任务**下发批量命令。

1. 应用调用**创建批量任务**接口，下发请求到物联网平台，下发消息样例如下。

```
POST https://{Endpoint}/v5/iot/{project_id}/batchtasks
Content-Type: application/json
```

```
X-Auth-Token: ********
{
  "app_id": "84fb64e43c5f4c6cbec339e52449bcea",
  "task_name": "task123",
  "task_type": "createCommands",
  "targets": [
    "5f2bc9b961e7670469c5ef6d_1997930",
    "5f2bc9b961e7670469c5ef6d_1997931"
  ],
  "document": {
    "service_id": "water",
    "command_name": "ON_OFF",
    "paras": {
      "value": "ON"
    }
  }
}
```

- app_id：资源空间ID
- task_name：任务名（自定义）
- task_type：createCommands
- targets：设备ID数组
- document：命令相关参数，参考**设备同步命令**

2. 物联网平台向应用返回“201 Created”。
3. 设备订阅下行topic接收命令，并通过上行topic向平台响应命令结果， 参考**平台命令下发**。
4. 通过调用**查询批量任务**接口查询批量命令下发任务执行情况。

相关 API 接口

- **下发设备消息**
- **修改设备属性**
- **查询设备属性**
- **下发设备命令**
- **查询设备消息**
- **平台命令下发**
- **平台消息下发**
- **查询指定消息id的消息**

4.6 订阅推送

4.6.1 订阅推送方式概述

设备接入到物联网平台后，便可与物联网平台进行通信。设备通过自定义Topic或产品模型方式将数据上报到平台，在控制台设置后，通过订阅推送的方式，将设备生命周期变更、设备属性上报、设备消息上报、设备消息状态变更、设备状态变更、批量任务状态变更等消息推送到您指定的服务器。

当前华为物联网平台支持HTTP/HTTPS和AMQP两种订阅方式：

- HTTP/HTTPS订阅推送：应用服务器通过调用物联网平台的**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**接口配置并激活规则，在指定应用服务器的

URL后，将平台获取发生变更的设备业务信息（如设备生命周期管理、设备数据上报、设备消息状态、设备状态等）和管理信息（软固件升级状态和升级结果）推送给指定URL的服务器。

- AMQP订阅推送：无需对接其他华为云服务，即可实现数据流转。应用服务器通过调用物联网平台的**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**接口配置并激活规则，在AMQP客户端和平台建链后，物联网平台根据用户订阅的数据类型，即可将对应的变更信息推送给指定的AMQP消息队列。详细请参考**使用AMQP订阅推送**。

订阅推送	适用场景	优点	限制
HTTP/HTTPS订阅推送	应用作为服务端被动接收IoT云服务的消息	/	/
AMQP订阅推送	应用作为客户端，可主动拉取IoT云服务的消息，也可以通过监听被动接收 IoT云服务的消息。	能主动拉取数据	请参考**连接规格**。

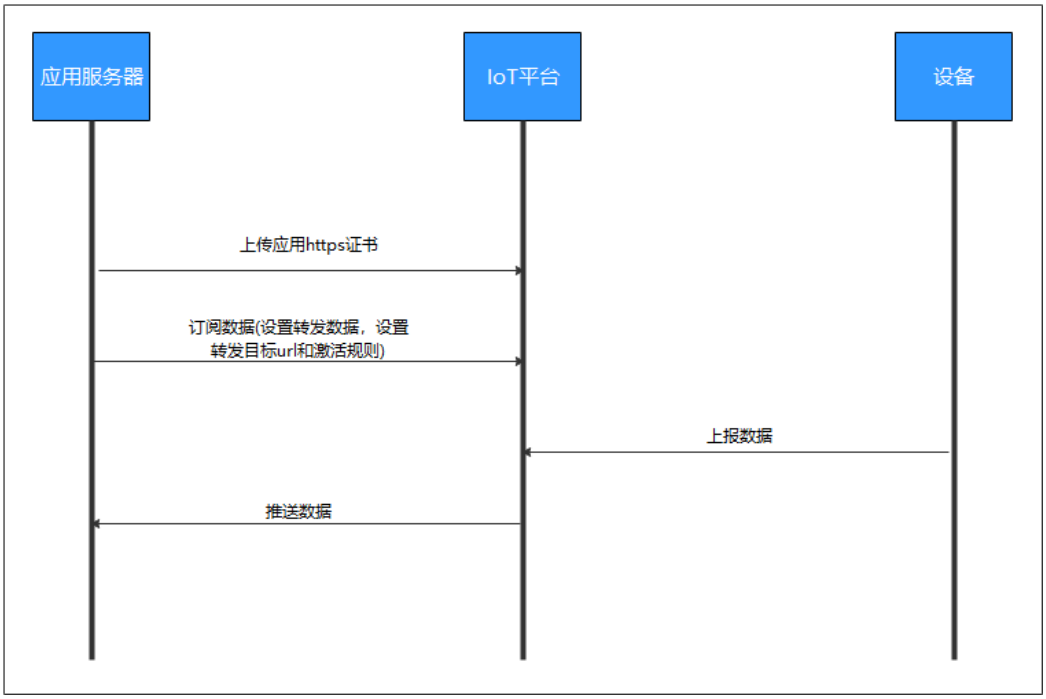
4.6.2 使用 HTTP/HTTPS 订阅推送

概述

订阅：是指应用服务器通过调用物联网平台的**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**接口配置并激活规则，或者在控制台创建订阅任务，向平台获取发生变更的设备业务信息（如设备生命周期管理、设备数据上报、设备消息状态、设备状态等）和管理信息（软固件升级状态和升级结果）。订阅时必须指定应用服务器的URL，也称为回调地址。（**什么是回调地址？**）。

推送：是指订阅成功后，物联网平台根据应用服务器订阅的数据类型，将对应的变更信息（推送的通知内容可参考**流转数据**）推送给指定的URL地址。如果应用服务器没有订阅该类型的数据通知，即使数据发生了变更也不会进行推送。物联网平台进行数据推送时，数据格式为JSON格式，推送协议可以采用HTTP或HTTPS协议，其中HTTPS协议为加密传输协议，需要进行安全认证，更加安全，推荐使用。

订阅推送的示意图如下图所示：



物联网平台采用HTTPS协议向应用服务器进行消息推送时，物联网平台需要校验应用服务器的真实性，需要在物联网平台上加载CA证书，该证书由应用服务器侧提供（调测时可自行**制作调测证书**，商用时建议更换为商用证书，否则会带来安全风险）。

推送机制：物联网平台向应用服务器推送消息后，如果应用服务器接收消息成功，会向物联网平台返回200 OK响应码。如果应用服务器无响应（或响应时间超过15秒）,或者应用服务器向物联网平台返回非200响应码(如500,501,502,503,504等)，表示消息推送失败，推送失败连续累计达到10次，物联网平台会将该订阅URL的主机地址加入黑名单保持5分钟，拉黑期间物联网平台对该主机地址暂停推送，5分钟后对该地址解除黑名单,重新进行推送。如果推送地址仍然无效，达到拉黑条件会再次触发黑名单机制；如果恢复，则会重新恢复正常推送。

如何进行数据订阅

应用服务器接入到“设备接入服务”后，在控制台创建订阅任务，也可以通过调用API接口进行数据订阅。

- 在控制台配置HTTP/HTTPS订阅请参考**配置HTTP/HTTPS服务端订阅**、**加载推送证书**。
- 通过API接口进行数据订阅请参考**如何调用API**和**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**。

推送数据格式

数据订阅成功后，物联网平台推送到应用侧的数据格式样例请参考**流转数据**。

说明

http协议消息头中，媒体类型信息Content-Type为application/json;字符集为utf-8;

加载推送证书

使用HTTPS协议推送，需要参考本小节完成推送证书的加载，然后参考**HTTP/HTTPS服务端订阅**在控制台创建订阅任务。

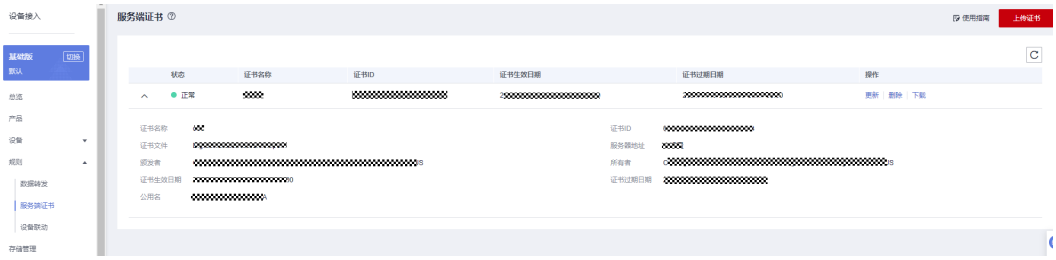
- 如果应用服务器取消了订阅后再重新订阅（URL不变），需要在物联网平台上重新上传CA证书。
- 如果应用服务器新增了订阅类型（新增URL），需要在物联网平台上加载与该URL对应的CA证书。即使新增URL使用的CA证书与原来推送的URL使用相同的证书，也需要重新上传CA证书。

步骤1 访问**设备接入服务**，单击“管理控制台”进入设备接入控制台。

步骤2 在左侧导航栏选择“规则 > 服务端证书”，单击右上角的“上传证书”，按照下表填写相关信息后，单击“确定”完成证书的加载。

参数名称	参数说明
证书名称	可自定义，用于区分不同证书
CA证书	需要提前申请和购买CA证书文件，CA证书由应用服务器侧提供。 **说明** 调测时可自行**制作调测证书**，商用时建议更换为商用证书，否则会带来安全风险。
域名/IP与端口	物联网平台推送消息到应用服务器的域名或IP地址与端口信息。填写为**创建规则动作**时URL中对应的域名或IP地址与端口信息。例如：推送URL为“https://www.example.com:8443/example/”，域名/IP与端口则为“www.example.com:8443”。

步骤3 单击左侧导航栏规则>服务端证书，选择对应证书，单击 可获取证书ID，用于后续**创建规则动作**时，作为参数使用。



----结束

制作调测证书

调测证书，又叫做自签名证书，用于客户端通过HTTPS访问服务端时进行安全认证。在物联网平台的使用中，可用于物联网平台向应用服务器采用HTTPS协议推送数据时，物联网平台认证应用服务器的合法性。本文以Windows环境为例，介绍通过Openssl工具制作调测证书的方法，生成的证书为PEM编码格式的证书，后缀为.cer。

常见的证书存储格式如下表所示。

存储格式	说明
DER	二进制编码，后缀名**.der/.cer/.crt**
PEM	BASE 64编码，后缀名**.pem/.cer/.crt**
JKS	Java的证书存储格式，后缀名**.jks**

说明

自签名证书仅用于调测阶段，在商用时，您需要向知名CA机构申请证书，否则可能会带来安全风险。

步骤1 在浏览器中访问**这里**，下载并安装OpenSSL工具。

步骤2 以管理员身份运行cmd命令行窗口。

步骤3 执行cd c:\openssl\bin（请替换为openssl实际安装路径），进入openssl命令视图。

步骤4 执行如下命令生成CA根证书私钥文件ca_private.key。

```
openssl genrsa -passout pass:123456 -aes256  -out ca_private.key 2048
```

- aes256：代表加密算法。
- passout pass：代表私钥密码。
- 2048：代表密钥长度。

步骤5 执行如下命令使用CA根证书私钥文件生成csr文件ca.csr，用于**6**生成CA根证书。

```
openssl req -passin pass:123456 -new -key ca_private.key -out ca.csr -subj "/C=CN/ST=GD/L=SZ/O=Huawei/OU=IoT/CN=CA"
```

如下信息您可以根据实际情况进行修改。

- C：代表国家，填写CN。
- ST：地区，如GD。
- L：城市，如SZ。
- O：组织，如Huawei。
- OU：组织单位，如IoT。
- CN：Common Name，填写为CA的组织名，如CA。

步骤6 执行如下命令生成CA根证书ca.cer。

```
openssl x509 -req -passin pass:123456 -in ca.csr -out ca.cer -signkey ca_private.key -CAcreateserial -days 3650
```

如下信息您可以根据实际情况进行修改。

- passin pass：必须与**4**中设置的私钥密码保持一致。
- days：代表证书有效期。

步骤7 执行如下命令生成应用服务器端私钥文件。

```
openssl genrsa -passout pass:123456 -aes256 -out server_private.key 2048
```

步骤8 执行如下命令生成应用服务器端csr文件，用于生成服务端证书。

```
openssl req -passin pass:123456 -new -key server_private.key -out server.csr -subj "/C=CN/ST=GD/L=SZ/O=Huawei/OU=IoT/CN=appserver.iot.com"
```

如下信息您可以根据实际情况进行修改。

- C：代表国家，填写CN。
- ST：地区，如GD。
- L：城市，如SZ。
- O：组织，如Huawei。
- OU：组织单位，如IoT。
- CN：Common Name，一般填写为应用服务器的域名或IP。

步骤9 通过CA私钥文件ca_private.key对服务端csr文件server.csr进行签名，生成服务端证书文件server.cer。

```
openssl x509 -req -passin pass:123456 -in server.csr -out server.cer -sha256 -CA ca.cer -CAkey ca_private.key -CAserial ca.srl -CAcreateserial -days 3650
```

步骤10 （可选）如果您需要.crt/.pem后缀的证书，可以根据如下命令进行转换。下面将以server.cer转为为server.crt为例进行说明，需要转换ca.cer证书时，请将命令中的server替换为ca。

```
openssl x509 -inform PEM -in server.cer -out server.crt
```

步骤11 在openssl安装目录的bin文件夹下，获取生成的CA证书（ca.cer/ca.crt/ca.pem）、应用服务器证书（server.cer/server.crt/server.pem）和私钥文件（server_private.key）。其中CA证书用于加载到物联网平台，应用服务器证书和私钥文件用于加载到应用服务器。

----结束

配置 HTTP/HTTPS 服务端订阅

本小节介绍如何在物联网平台配置HTTP/HTTPS服务端订阅。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择左侧导航栏的“规则 > 数据转发”，单击页面右上角的“创建规则”。

步骤3 参考下表填写参数后，单击“创建规则”。

参数名	参数说明
规则名称	创建的规则名称。
规则描述	对该规则的描述。

参数名	参数说明
数据来源	• 设备：将操作设备的信息，如设备添加、设备删除、设备更新设置为数据来源。当数据来源选择“设备”时，不支持快速配置。 • 设备属性：将归属在某个资源空间下的设备上报给平台的属性值设置为数据来源。单击右侧的“快速配置”勾选需要转发的产品、属性、服务等数据。 • 设备消息：将归属在某个资源空间下的设备上报给平台的消息设置为转发目标。单击右侧的“快速配置”，仅转发指定Topic的数据。选择所属产品，填写Topic名称。您可以使用在产品详情页面**自定义的Topic**，也可以使用**平台预置的Topic**。 • 设备消息状态：将设备和平台之间流转的设备消息状态变更设置为转发目标。设备消息状态详见**这里**。当数据来源选择“设备消息状态”，不支持快速配置。 • 设备状态：将归属在某个资源空间下的直连设备状态变更转发至其他服务。单击“快速配置”，您可以转发设备状态为“在线”、“离线”和“异常”的设备信息到其他服务。物联网平台直连设备状态详见**这里**。 • 批量任务：将批量任务状态的数据设置为数据来源。当数据来源选择“批量任务”时，不支持快速配置。 • 产品：将操作产品的信息，如产品添加、产品删除、产品更新设置为数据来源。当数据来源选择“产品”时，不支持快速配置。 • 设备异步命令状态：针对LwM2M/CoAP协议的设备，物联网平台支持下发异步命令给设备。将异步命令的状态变更设置为数据来源。物联网平台设备异步命令状态详见**这里**。当数据来源选择“设备异步命令状态”时，不支持快速配置。 • 运行日志：将MQTT设备的业务运行日志设置为数据来源。当数据来源选择“运行日志”时，不支持快速配置。
触发事件	选择数据来源后，对应修改触发事件。
资源空间	您可以选择单个资源空间或所有资源空间。当选择“所有资源空间”时，不支持快速配置。
SQL语句	您需要编辑处理消息数据的SQL，设置数据转发目的地。 单击“编辑SQL”，编写处理消息字段的SQL。 SQL编写方法，可参考**SQL语句**。 **说明** • 仅标准版实例和企业版实例支持SQL语句编辑，基础版实例不支持。 • 若使用快速配置，将自动生成查询语句。生成的查询语句将覆盖您之前编辑的SQL语句。

步骤4 在设置转发目标页面，单击“添加”，在弹出的页面中参考下表配置完参数后，单击“确认”。

参数名	参数说明
转发目标	选择“第三方应用服务（HTTP推送）”。

参数名	参数说明
推送URL	物联网平台推送消息到应用服务器的URL。例如，推送URL为“https://www.example.com:8443/example/”，则**加载推送证书**时“域名/IP与端口”为“www.example.com:8443”。 • 如果“推送URL”使用HTTP协议，不需要使用CA证书； • 如果“推送URL”使用HTTPS协议，需要上传CA证书，证书的上传可参考**加载推送证书**。

步骤5 完成完整的规则定义后，单击“启动规则”，实现数据转发至HTTP/HTTPS消息队列。

----结束

平台认证

作为服务端，应用侧如果需要认证IoT平台的身份，需要加载IoT平台的CA证书。请参考**资源**页面获取。

常见问题

订阅推送业务热点咨询问题如下，更多咨询问题请访问**查看更多**。

- **如何获取证书?**
- **调用订阅接口时，回调地址如何获取?**
- **回调地址可以使用域名吗?**
- **订阅后消息推送失败，例如提示503如何处理?**
- **为什么设备上报一条数据后应用服务器会收到多条推送?**
- **调用订阅接口时，提示回调地址不合法**
- **调用删除单个订阅接口时，subscriptionId如何获取**

相关 API 接口

创建规则动作

创建规则触发条件

修改规则触发条件

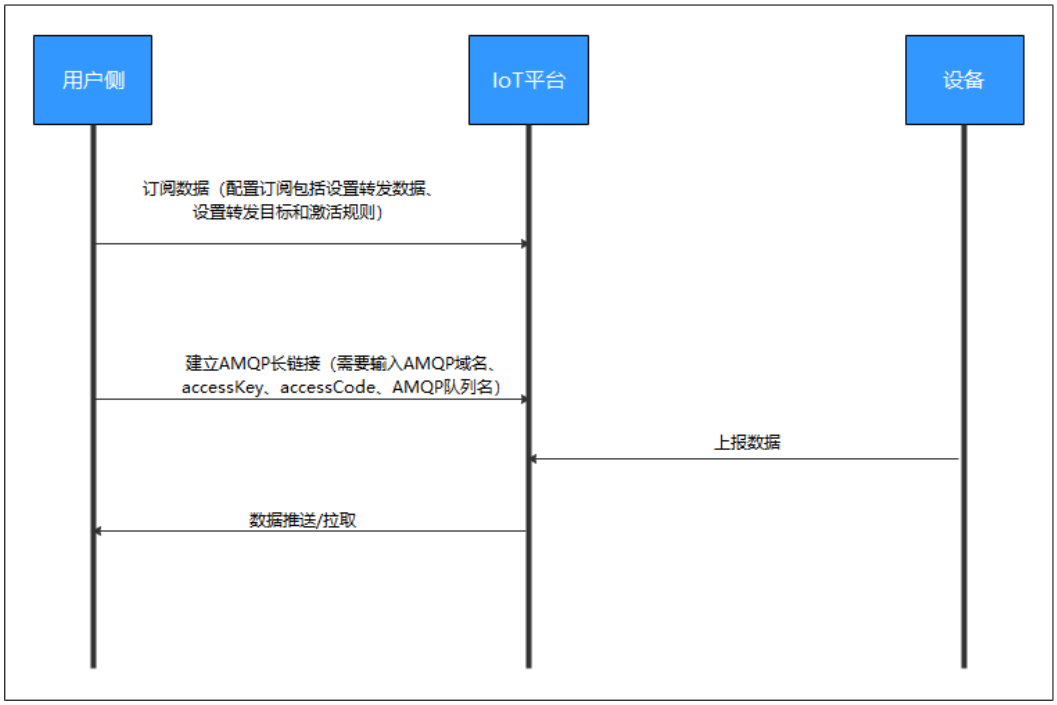
流转数据

4.6.3 使用 AMQP 订阅推送

4.6.3.1 AMQP 订阅推送

订阅：AMQP（Advanced Message Queuing Protocol）即高级队列消息协议。用户通过控制台创建订阅任务，也可以通过调用物联网平台的**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**接口配置并激活规则，向平台获取发生变更的设备业务信息（如设备生命周期管理、设备数据上报、设备消息状态、设备状态等）和管理信息（软固件升级状态和升级结果）。订阅时必须指定具体的AMQP消息通道。

推送：订阅成功后，物联网平台根据用户订阅的数据类型，将对应的变更信息推送给指定的AMQP消息队列。如果用户没有订阅该类型的数据通知，即使数据发生了变更也不会进行推送。用户可通过AMQP的客户端与IoT平台建立链接，来接收数据。订阅推送的示意图如下图所示：



推送机制：物联网平台向用户推送消息后，如果用户接收成功，会向物联网平台回复响应（推荐自动响应方式）。如果用户建立链接后不拉取数据，则会导致数据积压在服务端。服务端仅保存最近24小时，且占用磁盘容量小于1G的数据，如果用户不及时拉取数据，物联网平台会滚动清除超期和超出容量限制的数据。若用户收到消息后来不及响应，长链接中断，则未响应的数据会在下次链接后重新推送。

如何进行数据订阅

应用服务器接入到“设备接入服务”后，在控制台创建订阅任务，也可以通过调用API接口进行数据订阅。

- 在控制台配置AMQP订阅请参考**配置AMQP服务端订阅**。
- 通过API接口进行数据订阅请参考**如何调用API**和**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**。

推送数据格式

数据订阅成功后，物联网平台推送到应用侧的数据格式样例请参考**流转数据**。

说明

http协议消息头中，媒体类型信息Content-Type为application/json;字符集为utf-8;

相关 API 参考

创建规则动作

创建规则触发条件

修改规则触发条件

流转数据

创建AMQP队列

查询AMQP列表

查询单个AMQP队列

生成接入凭证

4.6.3.2 配置 AMQP 服务端订阅

本文介绍如何在物联网平台设置和管理AMQP服务端订阅。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择左侧导航栏的“规则 > 数据转发”，单击页面右上角的“创建规则”。

步骤3 参考下表填写参数后，单击“创建规则”。

参数名	参数说明
规则名称	创建的规则名称。
规则描述	对该规则的描述。

参数名	参数说明
数据来源	• 设备：将操作设备的信息，如设备添加、设备删除、设备更新设置为数据来源。当数据来源选择“设备”时，不支持快速配置。 • 设备属性：将归属在某个资源空间下的设备上报给平台的属性值设置为数据来源。单击右侧的“快速配置”勾选需要转发的产品、属性、服务等数据。 • 设备消息：将归属在某个资源空间下的设备上报给平台的消息设置为转发目标。单击右侧的“快速配置”，仅转发指定Topic的数据。选择所属产品，填写Topic名称。您可以使用在产品详情页面**自定义的Topic**，也可以使用**平台预置的Topic**。 • 设备消息状态：将设备和平台之间流转的设备消息状态变更设置为转发目标。设备消息状态详见**这里**。当数据来源选择“设备消息状态”，不支持快速配置。 • 设备状态：将归属在某个资源空间下的直连设备状态变更转发至其他服务。单击“快速配置”，您可以转发设备状态为“在线”、“离线”和“异常”的设备信息到其他服务。物联网平台直连设备状态详见**这里**。 • 批量任务：将批量任务状态的数据设置为数据来源。当数据来源选择“批量任务”时，不支持快速配置。 • 产品：将操作产品的信息，如产品添加、产品删除、产品更新设置为数据来源。当数据来源选择“产品”时，不支持快速配置。 • 设备异步命令状态：针对LwM2M/CoAP协议的设备，物联网平台支持下发异步命令给设备。将异步命令的状态变更设置为数据来源。物联网平台设备异步命令状态详见**这里**。当数据来源选择“设备异步命令状态”时，不支持快速配置。 • 运行日志：将MQTT设备的业务运行日志设置为数据来源。当数据来源选择“运行日志”时，不支持快速配置。
触发事件	选择数据来源后，对应修改触发事件。
资源空间	您可以选择单个资源空间或所有资源空间。当选择“所有资源空间”时，不支持快速配置。
SQL语句	您需要编辑处理消息数据的SQL，设置数据转发目的地。 单击“编辑SQL”，编写处理消息字段的SQL。 SQL编写方法，可参考**SQL语句**。 **说明** • 仅标准版实例和企业版实例支持SQL语句编辑，基础版实例不支持。 • 若使用快速配置，将自动生成查询语句。生成的查询语句将覆盖您之前编辑的SQL语句。

步骤4 在设置转发目标页面，单击“添加”，在弹出的页面中参考下表配置完参数后，单击“确认”。

参数名	参数说明
转发目标	选择“AMQP推送消息队列”

参数名	参数说明
消息队列	单击“选择”，选择消息队列。 • 若没有消息队列，请新建消息队列，队列名称自定义且单个租户名下唯一，最大长度 128位，支持大小写英文字符串、数字、下划线（ _ ）、中划线（ - ）和竖线（ | ），不支持斜杠（ / ）、空格等其他字符。 • 若需要删除消息队列，单击消息队列右侧的“删除”即可。 **说明** 已经订阅的队列不允许删除。

步骤5 完成完整的规则定义后，单击“启动规则”，实现数据转发至AMQP消息队列。

----结束

4.6.3.3 AMQP 客户端接入说明

在调用**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**配置并激活规则后，您需要参考本文将AMQP客户端接入物联网平台，成功接入后，在您的服务端运行AMQP客户端，即可接收订阅的消息。

协议版本说明

AMQP协议标准的详细介绍，请参见**AMQP协议标准**。

华为云物联网平台服务端订阅仅支持AMQP 1.0版的协议标准。

建连认证过程

1. AMQP客户端与物联网平台建立TCP连接，然后进行TLS握手校验。

   说明

   为了保障安全，接收方必须使用TLS加密，且使用TLS1.2及以上版本，不支持非加密的TCP传输。

2. 客户端请求建立连接。
3. 客户端向物联网平台发起请求，建立Receiver Link（即平台向客户端推送数据的单向通道）。

   客户端建立Connection成功后，需在15秒内完成Receiver Link的建立，否则物联网平台会关闭连接。

   建立Receiver Link后，客户端成功接入物联网平台。

   说明

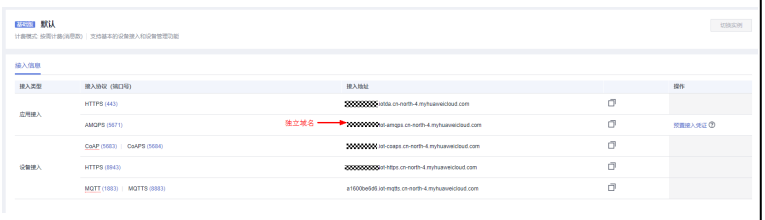
   一个Connection上最多能够创建十个Receiver Link，不支持创建Sender Link，即只能由平台向客户端推送消息，客户端不能向平台发送消息。

连接配置说明

AMQP客户端接入物联网平台的连接地址和连接认证参数说明如下：

- AMQP接入域名：amqps://${UUCID}.iot-amqps.cn-north-4.myhuaweicloud.com。

- 连接字符串：amqps://${UUCID}.iot-amqps.cn-north-4.myhuaweicloud.com :5671?
amqp.vhost=default&amqp.idleTimeout=8000&amqp.saslMechanisms=PLAIN

参数	说明
UUCID	独立域名ID（Unique User Connect ID），每个账号会自动生成，请前往**控制台**-总览页面-平台接入地址获取。 
amqp.vhost	当前amqp使用是默认的host，只支持default。
amqp.saslMechanisms	连接认证方式当前支持PLAIN-SASL。
idle-time-out	心跳时间单位为毫秒。如果超过心跳时间，Connection上没有任何帧通信，物联网平台将关闭连接。

- 端口：5671
- 客户端身份认证参数

  username =“accessKey=${accessKey}|timestamp=${timestamp}”

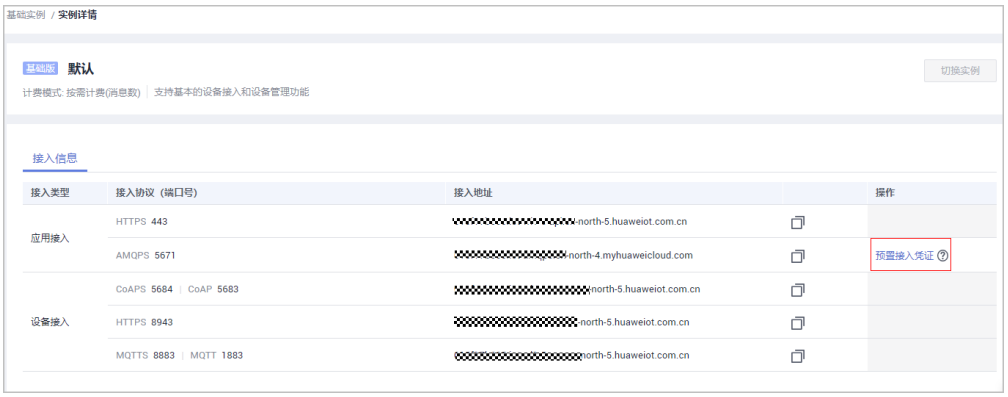
  password =“${accessCode}”

参数	是否必须	说明
accessKey	是	接入凭证键值，当前单个键值最多能和32个客户端同时进行建链。 首次建链时候，请参考**这里**进行预置。
timestamp	是	表示当前时间，13位毫秒值时间戳。 服务端校验客户端的时间戳，且时间戳相差5分钟。
accessCode	是	接入凭证密钥，长度不超过256个。 首次建链时候，请参考**这里**进行预置。若密钥丢失，可通过调用**接入凭证**接口进行重置，也可以参考**这里**进行重置。

获取 AMQP 接入凭证

若应用使用AMQP协议接入物联网平台进行数据流转，请先预置接入凭证。您可以通过调用**生成接入凭证**接口预置，也可以前往控制台页面进行预置，详细方法请参考如下操作：

1. 选择“IoTDA实例 > 基础版”，单击“详情”进入实例详情页面。
2. 单击“预置接入凭证”预置接入凭证密钥（accessCode）和接入凭证键值（accessKey）。



说明

如果您之前预置过接入凭证，重新预置之后，之前的接入凭证密钥将不能再使用。

连接规格

Key	Documentation
一个连接能够订阅queue数量	10
单个用户最大队列数	100
单个租户最大连接数	32
单个消息缓存时间	1天

接收平台推送的消息

客户端和平台之间的Receiver Link建链成功后，基于这个link，支持客户端主动拉取数据（推荐使用，客户端可根据自身能力来拉取数据）和注册监听由服务端推送两种方式。

4.6.3.4 Java SDK 接入示例

本文介绍使用AMQP协议的JMS客户端接入华为云物联网平台，接收服务端订阅消息的示例。

开发环境要求

本示例使用的开发环境为JDK 1.8及以上版本。

获取 SDK

AMQP SDK为开源SDK。如果您使用Java开发语言，推荐使用Apache Qpid JMS客户端。请访问**Qpid JMS 0.50.0**下载客户端和查看使用说明。

添加 Maven 依赖

```
<!-- amqp 1.0 qpid client -->
 <dependency>
   <groupId>org.apache.qpid</groupId>
```

```
  <artifactId>qpid-jms-client</artifactId>
  <version>0.61.0</version>
</dependency>
```

代码示例

您可以单击**这里**获取Java SDK接入示例，Demo中涉及的参数说明，请参考**AMQP客户端接入说明**。

⚠ 注意

所有示例代码已经包含与服务端断线重连的逻辑。

示例代码中用到的AmqpClient.java、AmqpClientOptions.java、AmqpConstants.java可以从**这里**获取。

1、创建AmqpClient。

```
// 以下参数请修改为自己的参数值
AmqpClientOptions options = AmqpClientOptions.builder()
                .host(AmqpConstants.HOST)
                .port(AmqpConstants.PORT)
                .accessKey(AmqpConstants.ACCESS_KEY)
                .accessCode(AmqpConstants.ACCESS_CODE)
                .queuePrefetch(1000) // sdk会在内存中分配该参数大小的队列，用来接收消息，客户端内存较小的情况可以调小该参数。
                .build();
        AmqpClient amqpClient = new AmqpClient(options);
        amqpClient.initialize();
```

2、通过设置listener消费amqp消息。

```
try {
    MessageConsumer consumer = amqpClient.newConsumer(AmqpConstants.DEFAULT_QUEUE);
    consumer.setMessageListener(message -> {
        try {
            // 此处进行消息处理。如果处理比较耗时，最好进行开启新的线程处理，否则可能造成心跳超时链接断开。
            processMessage(message.getBody(String.class));
            // 如果options.isAutoAcknowledge==false,此处应该调用message.acknowledge();
        } catch (Exception e) {
            log.warn("message.getBody error,exception is ", e);
        }
    });
} catch (Exception e) {
    log.warn("Consumer initialize error,", e);
}
```

3、主动拉取amqp消息

```
 // 创建一个线程池用来拉取消息
 ExecutorService executorService = new ThreadPoolExecutor(1, 1, 60, TimeUnit.SECONDS,new LinkedBlockingQueue<>(1));

    try {
        MessageConsumer consumer = amqpClient.newConsumer(AmqpConstants.DEFAULT_QUEUE);
        executorService.execute(() -> {
            while (!isClose.get()) {
                try {
                    Message message = consumer.receive();
                    // 此处进行消息处理。如果处理比较耗时，最好进行开启新的线程处理，否则可能造成心跳超时链接断开。
                    processMessage(message.getBody(String.class));
                    // 如果options.isAutoAcknowledge==false,此处应该调用message.acknowledge();
                } catch (JMSException e) {
```

```
                    log.warn("receive message error,", e);
                }
            }
        });
    } catch (Exception e) {
        log.warn("Consumer initialize error,", e);
    }
```

4、更多消费amqp消息的demo，请参考Java SDK接入示例工程。

资源

AmqpClient.java

```
package com.iot.amqp;

import lombok.extern.slf4j.Slf4j;
import org.apache.qpid.jms.JmsConnection;
import org.apache.qpid.jms.JmsConnectionExtensions;
import org.apache.qpid.jms.JmsConnectionFactory;
import org.apache.qpid.jms.JmsQueue;
import org.apache.qpid.jms.transports.TransportOptions;
import org.apache.qpid.jms.transports.TransportSupport;

import javax.jms.Connection;
import javax.jms.JMSException;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import java.util.Collections;
import java.util.HashSet;
import java.util.Set;

@Slf4j
public class AmqpClient {
    private final AmqpClientOptions options;
    private Connection connection;
    private Session session;
    private final Set<MessageConsumer> consumerSet = Collections.synchronizedSet(new HashSet<>());

    public AmqpClient(AmqpClientOptions options) {
        this.options = options;
    }

    public String getId() {
        return options.getClientId();
    }

    public void initialize() throws Exception {
        String connectionUrl = options.generateConnectUrl();
        log.info("connectionUrl={}", connectionUrl);
        JmsConnectionFactory cf = new JmsConnectionFactory(connectionUrl);
        // 信任服务端
        TransportOptions to = new TransportOptions();
        to.setTrustAll(true);
        cf.setSslContext(TransportSupport.createJdkSslContext(to));
        String userName = "accessKey=" + options.getAccessKey();
        cf.setExtension(JmsConnectionExtensions.USERNAME_OVERRIDE.toString(), (connection, uri) -> {
            // IoTDA的userName组成格式如下：“accessKey=${accessKey}|timestamp=${timestamp}”
            String newUserName = userName;
            if (connection instanceof JmsConnection) {
                newUserName = ((JmsConnection) connection).getUsername();
            }
            return newUserName + "|timestamp=" + System.currentTimeMillis();
        });
        // 创建连接
        connection = cf.createConnection(userName, options.getAccessCode());
        // 创建 Session, Session.CLIENT_ACKNOWLEDGE: 收到消息后，需要手动调用message.acknowledge()。
Session.AUTO_ACKNOWLEDGE: SDK自动ACK（推荐）。
        session = connection.createSession(false, options.isAutoAcknowledge() ?
```

```
Session.AUTO_ACKNOWLEDGE : Session.CLIENT_ACKNOWLEDGE);
        connection.start();
    }

    public MessageConsumer newConsumer(String queueName) throws Exception {
        if (connection == null || !(connection instanceof JmsConnection) || ((JmsConnection)
connection).isClosed()) {
            throw new Exception("create consumer failed,the connection is disconnected.");
        }
        MessageConsumer consumer;

        consumer = session.createConsumer(new JmsQueue(queueName));
        if (consumer != null) {
            consumerSet.add(consumer);
        }
        return consumer;
    }

    public void close() {
        consumerSet.forEach(consumer -> {
            try {
                consumer.close();
            } catch (JMSException e) {
                log.warn("consumer close error,exception is ", e);
            }
        });

        if (session != null) {
            try {
                session.close();
            } catch (JMSException e) {
                log.warn("session close error,exception is ", e);
            }
        }

        if (connection != null) {
            try {
                connection.close();
            } catch (JMSException e) {
                log.warn("connection close error,exception is", e);
            }
        }
    }
}
```

AmqpClientOptions.java

```
package com.iot.amqp;

import lombok.Builder;
import lombok.Data;
import org.apache.commons.lang3.StringUtils;

import java.text.MessageFormat;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;
import java.util.stream.Collectors;

@Data
@Builder
public class AmqpClientOptions {
    private String host;
    @Builder.Default
    private int port = 5671;
    private String accessKey;
    private String accessCode;
    private String clientId;
    /**
```

```
     * 仅支持true
     */
    @Builder.Default
    private boolean useSsl = true;

    /**
     * IoTDA仅支持default
     */
    @Builder.Default
    private String vhost = "default";

    /**
     * IoTDA仅支持PLAIN
     */
    @Builder.Default
    private String saslMechanisms = "PLAIN";

    /**
     * true: SDK自动ACK（默认）
     * false:收到消息后，需要手动调用message.acknowledge()
     */
    @Builder.Default
    private boolean isAutoAcknowledge = true;

    /**
     * 重连时延（ms）
     */
    @Builder.Default
    private long reconnectDelay = 3000L;

    /**
     * 最大重连时延（ms）,随着重连次数增加重连时延逐渐增加
     */
    @Builder.Default
    private long maxReconnectDelay = 30 * 1000L;

    /**
     * 最大重连次数,默认值-1，代表没有限制
     */
    @Builder.Default
    private long maxReconnectAttempts = -1;

    /**
     * 空闲超时，对端在这个时间段内没有发送AMQP帧则会导致连接断开。默认值为30000。单位：毫秒。
     */
    @Builder.Default
    private long idleTimeout = 30 * 1000L;

    /**
     * The values below control how many messages the remote peer can send to the client and be held in a
pre-fetch buffer for each consumer instance.
     */
    @Builder.Default
    private int queuePrefetch = 1000;

    /**
     * 扩展参数
     */
    private Map<String, String> extendedOptions;

    public String generateConnectUrl() {
        String uri = MessageFormat.format("{0}://{1}:{2}", (useSsl ? "amqps" : "amqp"), host,
String.valueOf(port));
        Map<String, String> uriOptions = new HashMap<>();
        uriOptions.put("amqp.vhost", vhost);
        uriOptions.put("amqp.idleTimeout", String.valueOf(idleTimeout));
        uriOptions.put("amqp.saslMechanisms", saslMechanisms);

        Map<String, String> jmsOptions = new HashMap<>();
```

```
        jmsOptions.put("jms.prefetchPolicy.queuePrefetch", String.valueOf(queuePrefetch));
        if (StringUtils.isNotEmpty(clientId)) {
            jmsOptions.put("jms.clientID", clientId);
        } else {
            jmsOptions.put("jms.clientID", UUID.randomUUID().toString());
        }
        jmsOptions.put("failover.reconnectDelay", String.valueOf(reconnectDelay));
        jmsOptions.put("failover.maxReconnectDelay", String.valueOf(maxReconnectDelay));
        if (maxReconnectAttempts > 0) {
            jmsOptions.put("failover.maxReconnectAttempts", String.valueOf(maxReconnectAttempts));
        }
        if (extendedOptions != null) {
            for (Map.Entry<String, String> option : extendedOptions.entrySet()) {
                if (option.getKey().startsWith("amqp.") || option.getKey().startsWith("transport.")) {
                    uriOptions.put(option.getKey(), option.getValue());
                } else {
                    jmsOptions.put(option.getKey(), option.getValue());
                }
            }
        }
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append(uriOptions.entrySet().stream()
                .map(option -> MessageFormat.format("{0}={1}", option.getKey(), option.getValue()))
                .collect(Collectors.joining("&", "failover:(" + uri + "?", ")")));
        stringBuilder.append(jmsOptions.entrySet().stream()
                .map(option -> MessageFormat.format("{0}={1}", option.getKey(), option.getValue()))
                .collect(Collectors.joining("&", "?", "")));
        return stringBuilder.toString();
    }
}
```

AmqpConstans.java

```
package com.iot.amqp;


public interface AmqpConstants {
    /**
     * AMQP接入域名
     * 参考：https://support.huaweicloud.com/usermanual-iothub/iot_01_00100_2.html#section2
     * eg: "****.iot-amqps.cn-north-4.myhuaweicloud.com";
     */
    String HOST = "127.0.0.1";

    /**
     * AMQP接入端口
     * 参考：https://support.huaweicloud.com/usermanual-iothub/iot_01_00100_2.html#section2
     */
    int PORT = 5671;

    /**
     * 接入凭证键值
     * 参考：https://support.huaweicloud.com/usermanual-iothub/iot_01_00100_2.html#section3
     * 不需要拼接时间戳timestamp
     */
    String ACCESS_KEY = "accessKey";

    /**
     * 接入凭证密钥
     * 参考：https://support.huaweicloud.com/usermanual-iothub/iot_01_00100_2.html#section3
     */
    String ACCESS_CODE = "accessCode";

    /**
     * 默认队列
     */
    String DEFAULT_QUEUE = "DefaultQueue";
}
```

4.6.3.5 Node.js SDK 接入示例

本文介绍使用Node.js语言的AMQP SDK接入华为云物联网平台，接收服务端订阅消息的示例。

开发环境

本示例所使用的开发环境为Node.js 8.0.0及以上版本。

下载 SDK

Node.js版本AMQP SDK，推荐使用rhea。请访问**rhea**下载库和查看使用说明。

添加依赖

在package.json中添加以下依赖。

```
"dependencies": {
    "rhea": "^1.0.12"
 }
```

代码示例

您可以单击**此处**获取Node.js SDK接入示例，Demo中涉及的参数说明，请参考**AMQP客户端接入说明**。

```
const container = require('rhea');
//获取当前时间戳
var timestamp = Math.round(new Date());

//建立连接。
var connection = container.connect({
    //接入域名，请参考这里。
    'host': '${UUCID}.iot-amqps.cn-north-4.myhuaweicloud.com',
    'port': 5671,
    'transport': 'tls',
    'reconnect': true,
    'idle_time_out': 8000,
    //userName组装方法，请参考这里。
    'username': 'accessKey=${yourAccessKey}|timestamp=' + timestamp + '|',
    //accessCode，请参考这里。
    'password': '${yourAccessCode}',
    'saslMechannisms': 'PLAIN',
    'rejectUnauthorized': false,
    'hostname': 'default',
});

//创建Receiver连接。 队列名，可以使用默认队列DefaultQueue
var receiver = connection.open_receiver('${yourQueue}');

//接收云端推送消息的回调函数。
container.on('message', function (context) {
    var msg = context.message;
    var content = msg.body;
    console.log(content);
    //发送ACK，注意不要在回调函数有耗时逻辑。
    context.delivery.accept();
 });
```

4.6.3.6 C# SDK 接入示例

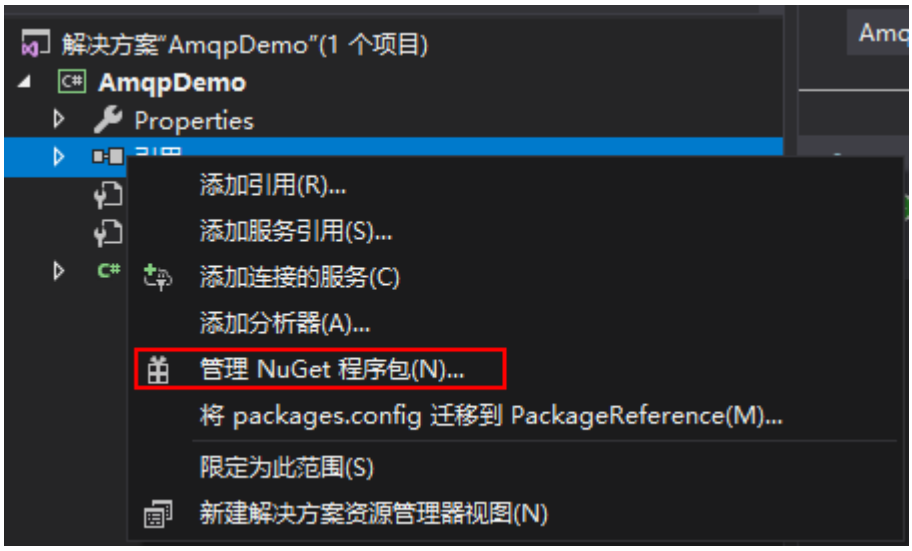
本文介绍使AMQPNetLite客户端接入华为云物联网平台，接收服务端订阅消息的示例。

开发环境要求

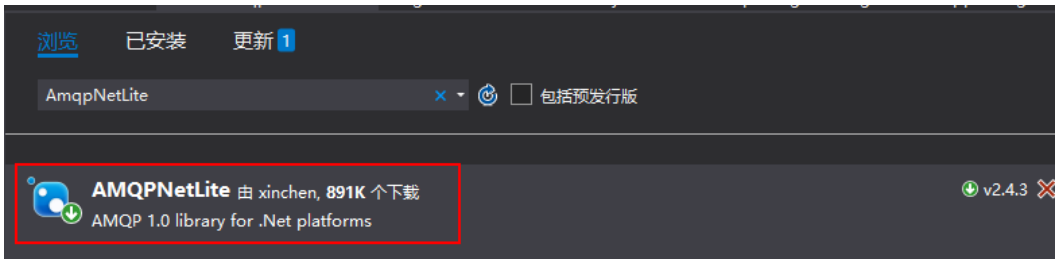
本示例使用的开发环境为.NETFramework V4.6及以上版本。

获取 SDK

1、在工程目录上点击鼠标右键打开"管理NuGet程序包"



2、在"NuGet管理器"中搜索到"AmqpNetLite"后安装所需版本



代码示例

Demo中涉及的参数说明，请参考**AMQP客户端接入说明**。

```
using Amqp;
using Amqp.Framing;
using Amqp.Sasl;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace AmqpDemo
{
    class Program
    {
        /// <summary>
        /// 接入域名，请参见AMQP客户端接入说明文档
        /// 请参考 https://support.huaweicloud.com/devg-iothub/iot_01_00100_2.html#section2
        /// </summary>
        static string Host = "${Host}";
```

```
/// <summary>
/// 端口
/// </summary>
static int Port = 5671;

/// <summary>
/// 接入凭证键值
/// </summary>
static string AccessKey = "${YourAccessKey}";

/// <summary>
/// 接入凭证密钥
/// </summary>
static string AccessCode = "${yourAccessCode}";

/// <summary>
/// 队列名
/// </summary>
static string QueueName = "${yourQueue}";

static Connection connection;

static Session session;

static ReceiverLink receiverLink;

static DateTime lastConnectTime = DateTime.Now;

static void Main(string[] args)
{
    try
    {
        var connection = CreateConnection();
        // 添加Connection Exception回调
        connection.AddClosedCallback(ConnectionClosed);

        // 创建Session。
        var session = new Session(connection);

        // 创建ReceiverLink
        var receiver = new ReceiverLink(session, "receiverName", QueueName);

        //接收消息。
        ReceiveMessage(receiver);
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
    }

    // 按下"Enter"键后程序退出
    Console.ReadLine();

    ShutDown();
}

/// <summary>
/// 创建Connection
/// </summary>
/// <returns>Connection</returns>
static Connection CreateConnection()
{
    lastConnectTime = DateTime.Now;
    long timestamp = new DateTimeOffset(DateTime.UtcNow).ToUnixTimeMilliseconds();
    string userName = "accessKey=" + AccessKey + "|timestamp=" + timestamp;
    Address address = new Address(Host, Port, userName, AccessCode);
    ConnectionFactory factory = new ConnectionFactory();
    factory.SASL.Profile = SaslProfile.External;
```

```
        // 信任服务端,跳过证书校验
        factory.SSL.RemoteCertificateValidationCallback = (sender, certificate, chain, sslPolicyError) =>
{ return true; };
        factory.AMQP.IdleTimeout = 8000;
        factory.AMQP.MaxFrameSize = 8 * 1024;
        factory.AMQP.HostName = "default";
        var connection = factory.CreateAsync(address).Result;
        return connection;
    }

    static void ReceiveMessage(ReceiverLink receiver)
    {
        receiver.Start(20, (link, message) =>
        {
            // 在线程池中处理消息，防止阻塞拉取消息的线程
            ThreadPool.QueueUserWorkItem((obj) => ProcessMessage(obj), message);
            // 回ACK
            link.Accept(message);
        });
    }

    static void ProcessMessage(Object obj)
    {
        if (obj is Message message)
        {
            string body = message.Body.ToString();
            Console.WriteLine("receive message, body=" + body);
        }
    }

    static void ConnectionClosed(IAmqpObject amqpObject, Error e)
    {
        // 断线重连,15s重连一次
        while (DateTime.Now.CompareTo(lastConnectTime.AddSeconds(15)) < 0)
        {
            Thread.Sleep(1000);
        }
        ShutDown();

        var connection = CreateConnection();
        // 添加Connection Exception回调
        connection.AddClosedCallback(ConnectionClosed);

        // 创建Session。
        var session = new Session(connection);

        // 创建ReceiverLink
        var receiver = new ReceiverLink(session, "receiverName", QueueName);

        //接收消息。
        ReceiveMessage(receiver);
    }

    static void ShutDown()
    {
        if (receiverLink != null)
        {
            try
            {
                receiverLink.Close();
            }
            catch (Exception e)
            {
                Console.WriteLine("close receiverLink error, exception =" + e);
            }

        }
        if (session != null)
        {
```

```
            try
            {
                session.Close();
            }
            catch (Exception e)
            {
                Console.WriteLine("close session error, exception =" + e);
            }

        }

        if (connection != null)
        {
            try
            {
                connection.Close();
            }
            catch (Exception e)
            {
                Console.WriteLine("close connection error, exception =" + e);
            }

        }
    }
  }
}
```

4.6.3.7 Android SDK 接入示例

本文介绍Android系统如何通过Amqp接入华为云物联网平台，接收服务端订阅消息的示例。

开发环境

本示例所使用的开发环境如下：

JDK1.8及以上

Android API 28 Platform及以上

添加依赖

在build.gradle中添加以下依赖。

```
dependencies: {
    implementation 'org.apache.qpid:proton-j:0.33.8'
}
```

代码示例

您可以单击**此处**获取Android SDK接入示例，Demo中涉及的参数说明，请参考**AMQP客户端接入说明**。

5 规则引擎

5.1 规则引擎介绍

规则引擎是指用户可以在物联网平台上可以对接入平台的设备设定相应的规则，在条件满足所设定的规则后，平台会触发相应的动作来满足用户需求。包含设备联动和数据转发两种类型。

- **设备联动**

  设备联动指通过条件触发，基于预设的规则，引发多设备的协同反应，实现设备联动、智能控制。例如，当执行动作为“发送通知”时，物联网平台对接华为云的**消息通知服务SMN**，进行主题消息的设置和下发。

- **数据转发**

  数据转发无缝与华为云其他服务对接，实现设备数据的存储、计算、分析的全栈服务。

云服务访问授权

物联网平台支持与华为云其它云服务进行对接。首次创建对接到DIS服务、OBS服务、Kafka服务、ROMA Connect服务、以及SMN服务的规则时，您需要进行云服务访问授权操作。

授权成功后可以通过数据转发功能将物联网平台的数据转发到您购买的华为云其他服务，或通过设备联动规则，下发命令控制设备。

同时设备接入服务将在**统一身份认证服务（IAM）**为您创建名为iotda_admin_trust 的委托，并默认绑定管理员角色。

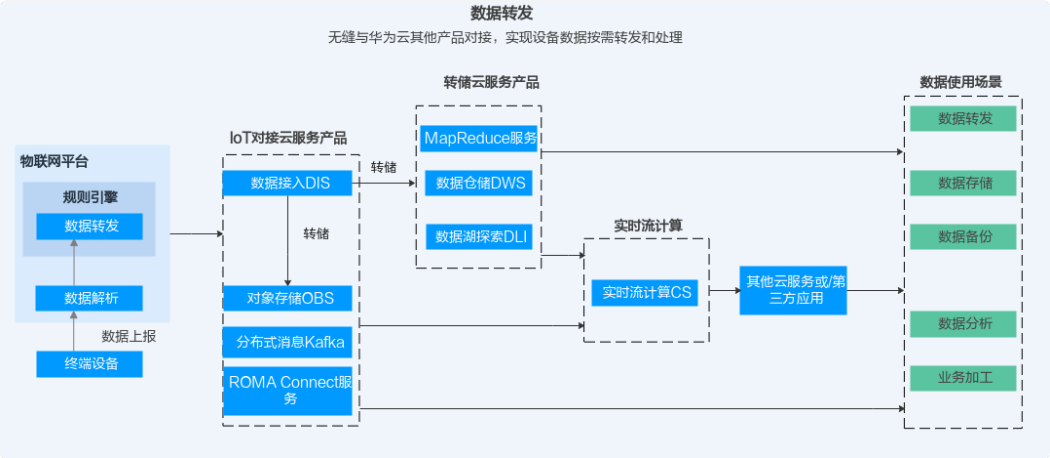
5.2 数据转发

概述

数据转发无缝与华为云其他服务对接，实现设备数据的存储、计算、分析的全栈服务。当前物联网平台支持转发至如下服务：

- **数据接入服务DIS**：实现数据高效采集、传输、分发。用户可以通过DIS提供的SDK/API等方式下载数据，完成后续自定义使用数据的业务开发场景；数据接入服务也可以通过转储任务进一步将数据转发到其他云服务（**OBS**、**MapReduce**、**DWS**、**DLI**），进行数据存储、数据分析等后续数据处理，便于用户进行更灵活、多样化的数据使用。
- **分布式消息服务Kafka**：为设备数据提供的消息队列服务。Kafka是一个拥有高吞吐、可持久化、可水平扩展，支持流式数据处理等多种特性的分布式消息流处理中间件，采用分布式消息发布与订阅机制，在日志收集、流式数据传输、在线/离线系统分析、实时监控等领域有广泛的应用。使用华为云分布式消息服务Kafka，资源按需申请，按需配置Topic的分区与副本数量，即买即用，您将有更多精力专注于业务快速开发，不用考虑部署和运维。
- **对象存储服务OBS**：实现设备数据持久存储。OBS是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力，适用于对设备上报数据进行归档和备份存储。OBS也支持对接**实时流计算CS云服务**，实时分析数据流，分析结果对接到其他云服务或者第三方应用进行数据可视化等。
- **应用与数据集成平台ROMA Connect**：它的消息集成（Message Queue Service，简称MQS）组件，为物联网平台与应用服务器之间提供安全、标准化的消息通道。MQS是一款企业级消息中间件，基于Kafka协议，使用统一的消息接入机制，并具备消息发布订阅、Topic管理、用户权限管理、资源统计、监控报警等基础功能，以及消息轨迹、网络隔离、云上云下集成等高级特性，为企业数据管理提供统一的消息通道。
- 第三方应用服务（HTTP推送）：应用服务器通过调用物联网平台的**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**接口配置并激活规则，在指定应用服务器的URL后，将平台获取发生变更的设备业务信息（如设备生命周期管理、设备数据上报、设备消息状态、设备状态等）和管理信息（软固件升级状态和升级结果）推送给指定URL的服务器。
- AMQP推送消息队列：无需对接其他华为云服务，即可实现数据流转。应用服务器通过调用物联网平台的**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**接口配置并激活规则，在AMQP客户端和平台建链后，物联网平台根据用户订阅的数据类型，即可将对应的变更信息推送给指定的AMQP消息队列。详细请参考**使用AMQP订阅推送**。
- 云日志服务（LTS）：实现日志采集、分析、查询、存储、可视化图表等功能。用户可以通过云日志服务快速高效地进行设备运维管理、用户业务趋势分析、安全监控审计等操作。应用服务器通过调用物联网平台的**创建规则触发条件**、**创建规则动作**、**修改规则触发条件**接口配置并激活规则，在指定日志组和日志流后，将MQTT设备的业务运行日志推送给指定的日志组和日志流。

图 5-1 数据转发



若您想要进一步体验数据转发功能，可参考**数据转发至OBS长期储存**。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择左侧导航栏的“规则 > 数据转发”，单击页面右上角的“创建规则”。

步骤3 设置转发数据，然后单击“创建规则”。

参数名	参数说明
规则名称	创建的规则名称。
规则描述	对该规则的描述。

参数名	参数说明
数据来源	• 设备：将操作设备的信息，如设备添加、设备删除、设备更新设置为数据来源。当数据来源选择“设备”时，不支持快速配置。 • 设备属性：将归属在某个资源空间下的设备上报给平台的属性值设置为数据来源。单击右侧的“快速配置”勾选需要转发的产品、属性、服务等数据。 • 设备消息：将归属在某个资源空间下的设备上报给平台的消息设置为转发目标。单击右侧的“快速配置”，仅转发指定Topic的数据。选择所属产品，填写Topic名称。您可以使用在产品详情页面**自定义的Topic**，也可以使用**平台预置的Topic**。 • 设备消息状态：将设备和平台之间流转的设备消息状态变更设置为转发目标。设备消息状态详见**这里**。当数据来源选择“设备消息状态”，不支持快速配置。 • 设备状态：将归属在某个资源空间下的直连设备状态变更转发至其他服务。单击“快速配置”，您可以转发设备状态为“在线”、“离线”和“异常”的设备信息到其他服务。物联网平台直连设备状态详见**这里**。 • 批量任务：将批量任务状态的数据设置为数据来源。当数据来源选择“批量任务”时，不支持快速配置。 • 产品：将操作产品的信息，如产品添加、产品删除、产品更新设置为数据来源。当数据来源选择“产品”时，不支持快速配置。 • 设备异步命令状态：针对LwM2M/CoAP协议的设备，物联网平台支持下发异步命令给设备。将异步命令的状态变更设置为数据来源。物联网平台设备异步命令状态详见**这里**。当数据来源选择“设备异步命令状态”时，不支持快速配置。 • 运行日志：将MQTT设备的业务运行日志设置为数据来源。当数据来源选择“运行日志”时，不支持快速配置。
触发事件	选择数据来源后，对应修改触发事件。
资源空间	您可以选择单个资源空间或所有资源空间。当选择“所有资源空间”时，不支持快速配置。
SQL语句	您需要编辑处理消息数据的SQL，设置数据转发目的地。 单击“编辑SQL”，编写处理消息字段的SQL。 SQL编写方法，可参考**SQL语句**。 **说明** • 仅标准版实例和企业版实例支持SQL语句编辑，基础版实例不支持。 • 若使用快速配置，将自动生成查询语句。生成的查询语句将覆盖您之前编辑的SQL语句。

步骤4 单击“设置转发目标”页签，单击“添加”，设置转发目标。

根据实际情况，选择**数据接入服务DIS**、**分布式消息服务Kafka**、**对象存储服务OBS**、**应用与数据集成平台ROMA Connect**、**第三方应用服务（HTTP推送）**、或**AMQP推送消息队列**或**云日志服务（LTS）**。

参数名	转发目标	参数说明
转发目标	数据接入服务（DIS）	• 区域：选择转发服务的所在区域，若未授权访问此区域的服务，请根据界面提示，配置云服务访问授权。 • 通道归属：可以选择自有通道或他人授权。 – 自有通道：选择需要的通道。若没有通道，请前往DIS服务**创建通道**。 – 他人授权：您也可以使用其他用户授权给您的通道。请前往DIS控制台获取通道ID。 **说明** 确保已在DIS控制台添加上传权限授权策略，详细操作请参考**DIS授权管理**。
	分布式消息服务（Kafka） **说明** 如果选择分布式消息服务，目前只支持转发到Kafka专享版。并且需要开启“kafka自动创建Topic”功能。	• 区域：选择转发服务的所在区域，若未授权访问此区域的服务，请根据界面提示，配置云服务访问授权。 • 对接地址：参考**连接已开启SASL的Kafka专享版实例**获取对接地址。设备接入基础版和标准版实例只支持公网接入专享版Kafka，企业版实例支持私网接入专享版Kafka。 • 主题：自定义。 • 数据加密传输：若开启数据加密传输，SASL用户名和密码，填写您在**购买Kafka实例**中输入的SASL用户名和密码。
	对象存储服务（OBS）	• 区域：选择转发服务的所在区域，若未授权访问此区域的服务，请根据界面提示，配置云服务访问授权。 • 存储桶：选择需要的桶。若没有，请前往OBS服务**创建桶**。 • 参数说明：OBS服务中存储通道文件的自定义目录,多级目录可用(/)进行分隔，不可以斜杠(/)开头或结尾，不能包含两个以上相邻的斜杠(/)。
	应用与数据集成平台（ROMA Connect） **说明** 目前仅企业版实例支持转发到ROMA Connect，基础版实例和标准版实例不支持。	• 对接地址：输入MQS的连接地址。详细指导请参考**这里**。 • 账户名/密码：访问ROMA实例的控制台获取MQS的用户名与密码。详细请参考**这里**。 • 主题：自定义。
	第三方应用服务（HTTP推送）	您可以使用HTTP或HTTPS协议进行推送。详细参数填写请参考**HTTP/HTTPS服务端订阅**。

参数名	转发目标	参数说明
	AMQP推送消息队列	消息队列：选择需要推送消息的队列。若没有队列，请创建队列。消息队列名限制请参考**配置AMQP服务端订阅**。
	函数工作流服务（FunctionGraph） **说明** 目前仅企业版实例支持转发到函数工作流服务，基础版实例和标准版实例不支持。	• 函数名称：选择需要调用的函数名称（latest版本）。当前不支持跨区域调用函数。若没有函数，请参考**这里**创建函数。

步骤5 启动规则。

完成完整的规则定义后，您可以单击“启动规则”，实现数据转发。

----结束

相关 API 参考

创建规则动作

创建规则触发条件

修改规则触发条件

流转数据

相关推荐

相关文档	相关产品
• **数据上报** • **使用HTTP/HTTPS订阅推送** • **使用AMQP订阅推送** • **查看运行日志**	• **消息通知服务SMN** • **数据接入服务DIS** • **分布式消息服务Kafka** • **对象存储服务OBS** • **应用与数据集成平台ROMA Connect** • **云日志服务（LTS）**

5.3 设备联动

概述

设备联动指通过条件触发，基于预设的规则，引发多设备的协同反应，实现设备联动、智能控制。例如：设置水表的电池电量阈值为小于等于20%时，上报电池电量过低的告警，用户就能及时了解设备的供电情况，以便及时更换电池。

若您想要进一步体验设备联动功能，可参考**设备触发告警并邮件或短信通知**。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择左侧导航栏的“规则 > 设备联动”，单击页面右上角的“创建规则”。

步骤3 参考下表参数说明，创建设备联动规则。

参数名称	说明	最佳实践
规则名称	创建的规则名称。	• **温度过高时自动关闭设备** • **设备触发告警并邮件或短信通知**
立即触发	平台创建规则前最后一次上报的数据满足触发条件，可以设置是否立即触发该规则。 • 勾选：创建规则后规则立即生效，系统会对最近一次上报的数据做判断来触发规则。 • 不勾选：创建规则后，系统对新上报的数据进行判断，然后触发规则。	
生效时间	• 一直生效：没有时间限制，持续检查当前规则条件是否满足。 • 指定时间：可以选择时间段，在特定的时间检查规则条件是否满足。	
描述	对该规则的描述。	

参数名称	说明	最佳实践
触发条件	满足条件：可设置满足全部条件，或者任意一个条件，触发设备。 • **匹配设备触发**：对使用相同产品模型的设备进行条件设置。 – 选择产品：选择特定产品。 – 选择服务：选择产品后选择对应的服务类型。 – 选择属性：选择满足条件后上报数据的设备属性。 **说明** ▪ 属性的数据类型为int和decimal时，支持选择多种判断符号。 ▪ 属性的数据类型为string，date time，jsonObject时，判断条件仅支持相等。 – 触发机制：选择是否需要触发策略选择重复抑制，建议选择需要。 – 数据时效（秒）：数据有效时间。例如：设备产生数据时间为19:00，时效设为30分钟，平台收到数据时间为20:00，该情况下即使满足触发条件也不触发动作。 • **指定设备触发**：对指定设备进行条件设置。请选择满足条件后上报数据的设备，服务以及属性，详细可参考“匹配设备触发”的参数。 • **定时触发**：可以设置规则触发的时间点，该条件一般用于周期性的触发条件，如每天7:00，关闭路灯。 **说明** 当触发条件选择“定时触发”，则动作中不能设置“发送通知”、“上报告警”、“恢复告警”的执行动作。 – 时间点：可以选择规则触发的起始时间点。 – 重复次数：规则重复触发的次数（1~1440次）。 – 间隔（分钟）：在起始时间点后，重复触发规则的时间间隔（1~1440分钟）。	
执行动作	单击“添加动作”，设置在规则触发后，需要执行的动作。 • **下发命令**：依次选择需要执行下发命令的设备、服务、命令，然后配置下发命令的参数。 • **发送通知**：选择SMN云服务对应所在区域。如果未授权，根据界面提示配置云服务访问授权，单击相应链接跳转到SMN云服务页面设置主题。 • **上报告警**：定义告警级别、告警名称、告警内容等。当触发设置的条件后，在**应用运维管理**界面产生一条对应的设备告警。 • **恢复告警**：定义告警级别、告警名称、告警内容等，表示清除设备上报到平台的告警。	

步骤4 单击右下角“创建规则”，完成设备联动规则的创建。新创建的规则默认状态为“激活”，您可以在规则列表“状态”列，禁用规则。

----结束

相关 API 参考

查询规则列表

创建设备数据规则（单个设备）

创建定时规则（重复触发）

创建定时规则（按天触发）

查询规则

修改规则

删除规则

修改规则状态

5.4 SQL 语句

创建数据转发规则时，需要编写SQL来解析和处理设备上报的JSON数据，JSON数据具体格式参考**流转数据**。二进制格式的数据不做解析，直接透传。本文主要介绍如何编写数据转发规则的SQL表达式。

SQL 语句

SQL语句由SELECT子句和WHERE子句组成，每个子句不能大于500个字符，暂不支持中文等其他字符集。SELECT子句和WHERE子句里的内容大小写敏感，SELECT和WHERE，AS等关键字大小写不敏感。

以设备消息上报为SQL源数据示例:

```
{
  "resource" : "device.message",
  "event" : "report",
  "event_time" : "20151212T121212Z",
  "notify_data" : {
    "header" : {
      "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
      "product_id" : "ABC123456789",
      "app_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
      "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
      "node_id" : "ABC123456789",
      "tags" : [ {
        "tag_value" : "testTagValue",
        "tag_key" : "testTagName"
      } ]
    },
    "body" : {
      "topic" : "topic",
      "content" : {
          "temperature" : 40,
          "humidty" : 24
      }
    }
```

```
 }
}
```

在源数据中，body中的content是设备消息上报的数据，设置当设备上报数据中temperature大于38时触发条件，并筛选出device_id、content，不需要任何其他字段时，SQL语句示例如下：

```
SELECT notify_data.header.device_id AS device_id, notify_data.body.content WHERE
notify_data.body.content.temperature > 38
```

当设备上报消息中temperature大于38度时，会触发转发，转发后的数据格式如下：

```
{
   "device_id": "d4922d8a-6c8e-4396-852c-164aefa6638f",
   "notify_data.body.content" : {
         "temperature" : 40,
         "humidty" : 24
    }
}
```

SELECT 子句

SELECT子句由SELECT后跟多个SELECT子表达式组成，子表达式可以为*，JSON变量，字符串常量或整数常量。JSON变量后跟一个AS关键字和AS变量，长度不超过32个字符。如果使用常量，则必须使用AS指定名称。

- JSON变量

  JSON变量支持大小写字母，数字，下划线和中划线，为了和减号的意思区分，当使用中划线的时候，请将JSON变量用双引号括起来，如："msg-type"。

  Json变量抽取嵌套结构体的数据

  ```
  {
   "a":"b",
   "c": {
     "d" : "e"
   }
  }
  ```

  c.d 即可抽取出字符串e，可以多层嵌套。

- AS变量

  AS变量由大小写字母组成，大小写敏感。目前支持[a-zA-Z_-]*的模式，如果使用中划线，需要使用双引号括起来。

- 常数整数

  正如标准的SQL一样，SELECT上面支持常数整数，常数后必须跟AS子句，如

  ```
  SELECT 5 AS number
  ```

- 常数字符串

  正如标准的SQL一样，SELECT上面支持常数字符串,目前支持[a-zA-Z_-]*的模式，需要使用单引号括起来，常数后必须跟AS子句，如

  ```
  SELECT 'constant_info' AS str
  ```

WHERE

在WHERE子句中，您可以用JSON变量进行布尔运算，进行一些非空判断，然后使用AND， OR关键字把结果组合起来。

- **为空判断 IS NULL， IS NOT NULL**

  为空判断可以用在WHERE子句中，如果JSON变量抽取不到数据，或者抽取到的数组为空，那么IS NULL成立，反之IS NOT NULL成立。

```
WHERE data IS NULL
```

- **大于小于运算符 > <**

  大于小于运算符可以用于WHERE子句中，当且仅当JSON变量的值为常量整数时，可以进行两个JSON变量的比较或者JSON变量和常量的比较。大于小于运算符也可以用于常量和常量的比较。也可以通过AND或者OR来连接起来运算

  比如

  ```
  WHERE data.number > 5 可以抽取出json表达式大于5的信息
  WHERE data.tag < 4 可以抽取出json表达式中小于4的信息
  WHERE data.number > 5 AND data.tag < 4 可以抽取出json表达式data.number大于5的信息并且json表达式data.tag中小于4的信息
  ```

- **=**

  =运算符可以用于WHERE子句中，用于JSON变量和JSON变量的比较、JSON变量整数和整数常量的比较、JSON变量字符串和字符串常量的比较。如果两个JSON变量IS NULL成立，那么=比较结果为false。也可以通过AND或者OR来连接起来运算

  ```
  WHERE data.number = 5 可以抽取出json表达式等于5的信息
  WHERE data.tag = 4 可以抽取出json表达式中等于4的信息
  WHERE data.number = 5 OR data.tag = 4 可以抽取出json表达式data.number等于5的信息或者json表达式data.tag中等于4的信息
  ```

调试 SQL 语句

物联网平台提供SQL在线调试功能。调试方法如下。

1. 编写SQL后，单击“调试语句”。
2. 在SQL调试对话框的调试参数页签下，输入用于调试数据，然后单击“启动调试”。

函数列表

规则引擎提供多种函数，您可以在编写SQL时使用这些函数，实现多样化数据处理。

函数名称	携带参数	用途	返回值类型	限制
GET_TAG	String tagKey	获取tag对应的value	字符串	-
CONTAINS_TAG	String tagKey	判断是否包含该tag	布尔值	-
GET_SERVICE	String serviceId	获取指定serviceId的service如果您在一个消息体里有多个serviceId相同的service，结果目前不保证。	Json结构体格式	只能在属性上报时使用

函数名称	携带参数	用途	返回值类型	限制
GET_SERVICES	String serviceId	获取指定serviceId的service如果您在一个消息体里有多个serviceId相同的service，结果目前不保证。	JSON数组格式	只能在属性上报时使用
CONTAINS_SERVICES	String serviceId	获取指定serviceId的service,将结果汇合为一个数组	布尔值	只能在属性上报时使用
GET_SERVICE_PROPERTIES	String serviceId	获取指定serviceId的service中的properties字段	Json结构体格式	只能在属性上报时使用
GET_SERVICE_PROPERTY	String serviceId, String propertyKey	获取指定serviceId的service中的properties中propertyKey的属性值	字符串格式	限制只能在属性上报时使用

5.5 连通性测试

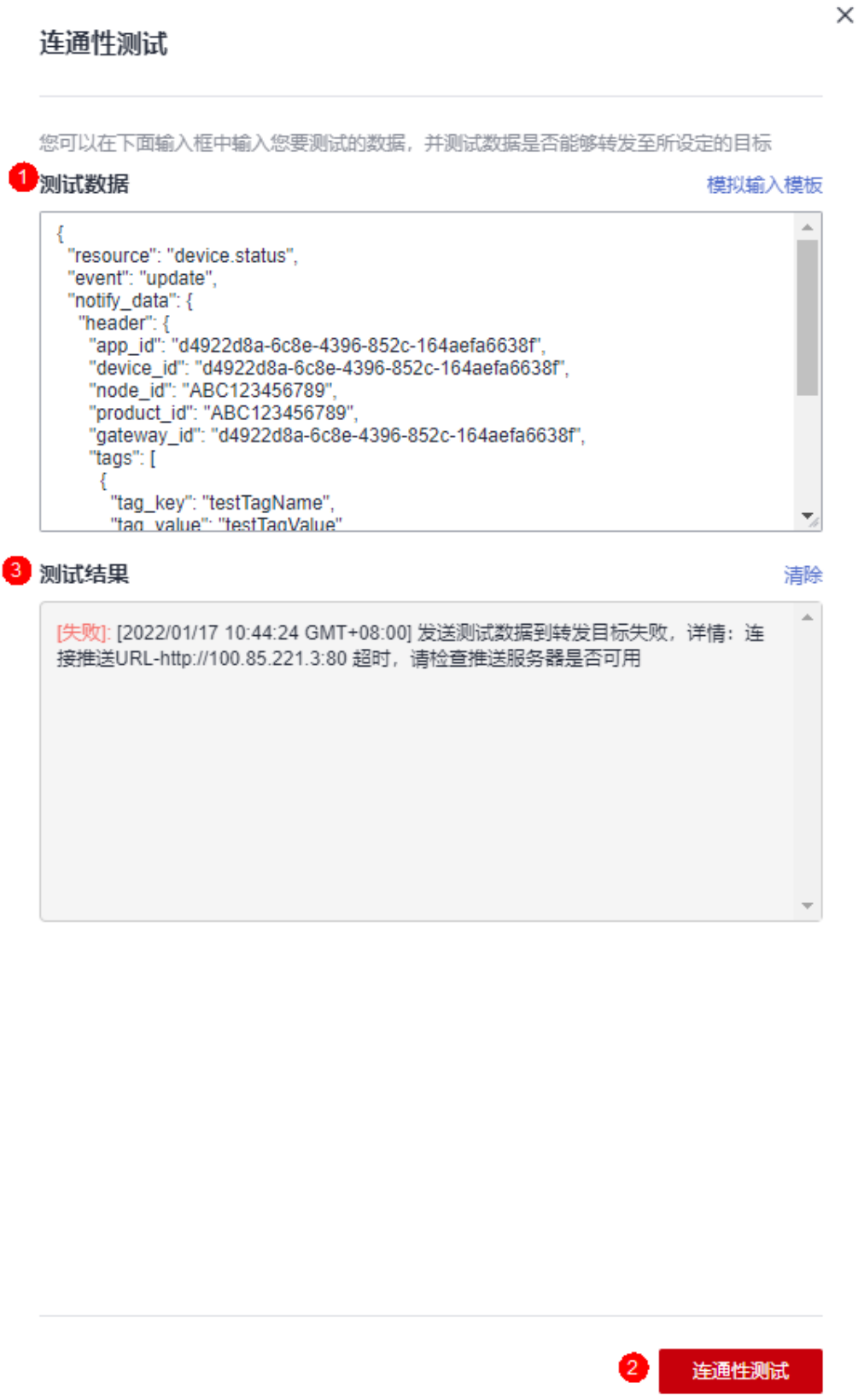
概述

物联网平台提供规则动作转发目标连通性测试功能。在业务对接调测阶段，您可使用连通性测试功能模拟业务数据调测规则动作的可用性及转发数据的业务一致性；在业务运行阶段数据转发出现故障时，您可使用连通性测试功能进行简单的问题复现及定位。

使用步骤

1、创建转发规则及动作后，单击待调测转发目标中的"测试"；

2、在连通性测试对话框的测试数据页签下，输入用于转发的测试数据，或单击右上角"模拟输入模板"，使用模板数据，然后单击"连通性测试"。



6 监控运维

6.1 设备消息跟踪

在设备鉴权、命令下发、数据上报、平台数据转发等业务场景中出现故障时，物联网平台可以通过消息跟踪功能进行快速的故障定位和原因分析。目前物联网平台支持NB-IoT设备和MQTT设备的消息跟踪，单个用户下，最多支持同时进行跟踪的设备数上限为10。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏选择“设备 > 所有设备”界面。

步骤3 通过检索条件，快速找到需要跟踪的设备，并单击"查看"按钮，进入设备详情。

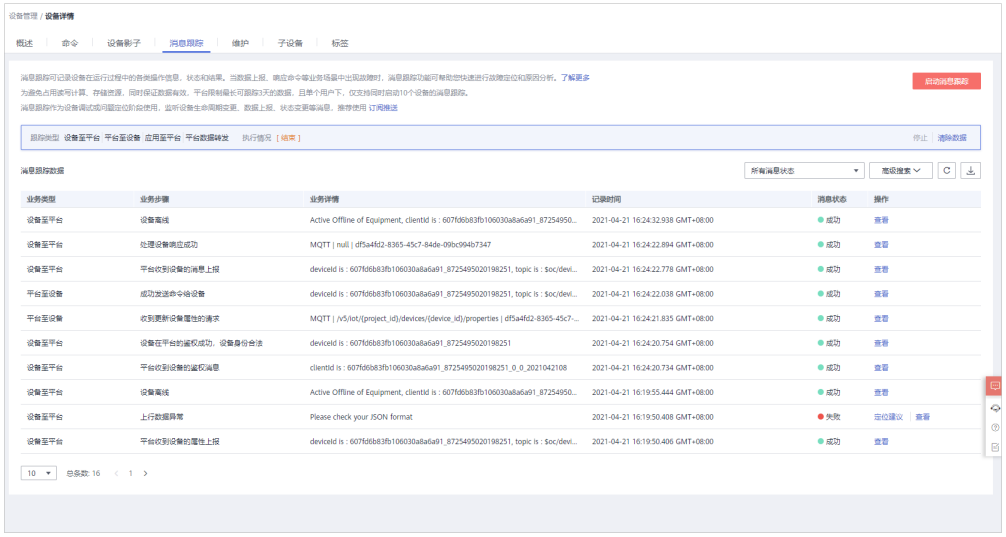
步骤4 在“设备详情”页，单击“消息跟踪”页签，单击“启动消息跟踪”按钮，选择需要跟踪的消息类型，并设置设备的消息跟踪时间，如下图所示，消息跟踪时长表示从启动消息跟踪功能开始到结束消息跟踪的总时长，在设置的时间段内进行消息跟踪，修改跟踪配置后，会已修改后时间为准。

图 6-1 启动消息跟踪

步骤5 您可以在“设备详情 > 消息跟踪”页面，查看正在进行消息跟踪的业务（如果单击停止消息跟踪按钮，则停止跟踪设备）。

如果跟踪的结果中数据量较多时，您可以根据“消息状态”、“消息类型”和“记录时间”进行过滤；如果对跟踪结果数据需要做进一步的分析，用户可以通过“导出”功能，将跟踪的结果数据导出，以便物联网平台运维人员做进一步的分析。

图 6-2 查看消息跟踪数据



若消息状态显示“失败”，可以通过单击“定位建议”按钮，查看失败场景的定位建议并进行问题的定位分析，通过单击“查看”按钮查看跟踪结果的详情信息。

----结束

6.2 查看报表

物联网平台为用户提供了丰富的报表功能，能够将数据直观地呈现出来，同时支持单击页面上方的资源空间切换按钮，查看不同资源空间下的统计数据。

在控制台左侧导航栏，选择“监控运维 > 统计报表”进入报表页面，报表名称及功能详细说明如下表所示。

报表名称	报表说明	数据刷新频率
设备状态	统计当前资源空间下，设备的在线率、未激活率、离线率和异常率。	每分钟
设备消息	设备上报消息数：统计每日设备上报的消息数，可查看上报数据趋势和平均消息上报速率（次数/秒）。	每小时
	下发命令消息数：统计每日平台下发的命令数。	每小时
设备总趋势	统计设备总数和设备在线数的趋势。	每小时
设备在线趋势	统计设备在线和离线情况，以百分比形式呈现。	每小时
设备数量统计	统计未激活设备数、异常设备数和离线设备数。	每小时
软件升级状态	统计当前资源空间下，历史上所有设备的软件升级任务，执行成功和失败的设备数量。	实时
固件升级状态	统计当前资源空间下，历史上所有设备的固件升级任务，执行成功和失败的设备数量。	实时
设备配置状态	统计当前资源空间下，历史上所有设备的设备配置更新任务，执行成功和失败的设备数量。	实时

更多报表，可登录**应用运维管理**控制台，选择监控 >指标浏览 >自定义指标，查看报表信息。目前，应用运维管理支持查看以下3类设备接入服务的监控信息：

报表名称	报表说明	数据刷新频率
IoTDA.Northbound Messages Pushed	统计应用侧消息的推送总次数、成功次数和失败次数。统计的数据为单位时间内的实时消息推送次数。	每分钟
IoTDA.Commands Issued	统计给设备下发的命令总数、成功数和失败数。统计的数据为单位时间内的实时数据	每分钟

报表名称	报表说明	数据刷新频率
IoTDA.Rules	统计创建的规则总数和单位时间内触发的规则总数。	每分钟

6.3 告警管理

当物联网平台监控到满足用户通过规则设置的告警触发条件时或设备消息上行的速度超过平台预设的阈值，平台就会上报告警。用户需要密切关注告警并及时进行处理，确保设备的正常运行。

告警分为规则类告警（暂未支持查看）和设备流控类告警。

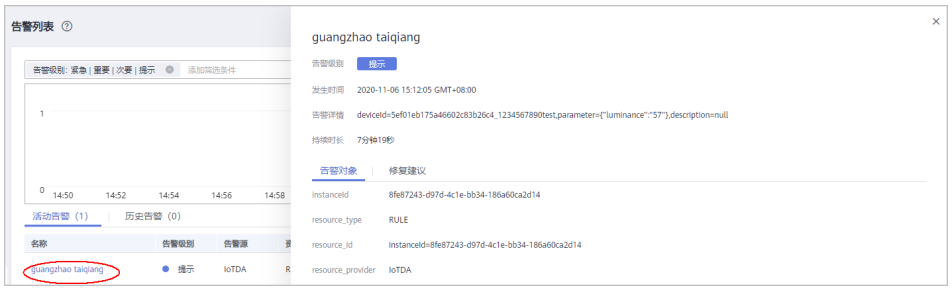
- 规则类告警：如果用户在控制台上设置设备联动类的**规则引擎**时，定义了响应动作为上报告警，且定义了告警属性、告警级别等，则当满足触发条件时，平台就会上报告警。例如：智能水表设备3天未上报数据，可能存在水表设备发生故障导致，平台会产生对应的告警，维护人员可通过告警信息找到对应告警的水表设备，进行快速定位维修。
- 设备流控类告警：平台监测到设备上行消息超过阈值时，会进入设备消息流控状态，平台会上报告警，包括以下4类：

告警名称	告警解释
设备上行消息流控告警	单个资源空间下的所有设备消息上报平台的速率之和超过设定的阈值（应用流控总开关默认关闭，开启后流控默认值为120条/秒），会进入设备消息流控状态，平台会上报告警。
单设备COAP消息流控告警	当平台统计周期内接收的单个NB-IoT设备消息数超过设定的阈值（默认值为300条/分钟），会进入单设备流控状态，平台会上报告警。
单设备COAP消息发包速率过高的告警	当NB-IoT设备发包次数在一定时间内超过阈值（默认值为5条/秒），触发告警同时会拒绝设备请求。
单设备MQTT消息流控告警	单个MQTT设备连接每秒上行数据的流量大小超过设定的阈值（默认值为3K/秒）时，会进入设备连接流控状态，平台会上报告警。

- 查看告警信息

  用户可以使用应用运维管理服务，查看告警。该服务支持查看最近30天的告警信息。

  a. 访问**设备接入服务**，单击“管理控制台”，进入设备接入控制台。
  b. 在左侧导航栏选择“监控运维>设备告警”，单击“前往查看”，跳转至应用运维管理服务，查看设备接入服务的告警信息。
  c. 单击目标告警，查看告警详情。



d. 清除告警。当设备故障解除时，可在告警列表中单击目标告警所在“操作”列中的 。

更多详细内容，请参考**查看告警**。

6.4 查看审计日志

用户在使用物联网平台的过程中，通过云审计服务，可查看用户及平台的操作及结果，当某项功能发生异常时，用户可以根据日志的记录信息定位并处理故障信息。目前支持查看以下设备接入服务的操作记录：

操作名称	资源类型	事件名称
创建规则	rule	createRule
删除规则	rule	deleteRule
更新规则	rule	updateRule
修改规则状态	rule	changeRuleStatus
下发设备命令	command	sendCommand
创建批量任务	batchtasks	createBatchTask
上传证书	certificate	addCertificate
删除证书	certificate	deleteCertificate
验证证书	certificate	verifyCertificate
注册设备	device	addDevice
删除设备	device	deleteDevice
重置设备密钥	device	resetDeviceSecret
添加设备组	device-group	addDeviceGroup
修改设备组	device-group	updateDeviceGroup
删除设备组	device-group	deleteDeviceGroup
管理设备组中的设备	device-group	manageDevicesInGroup
配置设备影子预期数据	deviceShadow	updateDeviceShadow

操作步骤

步骤1 登录**云审计服务**管理控制台。

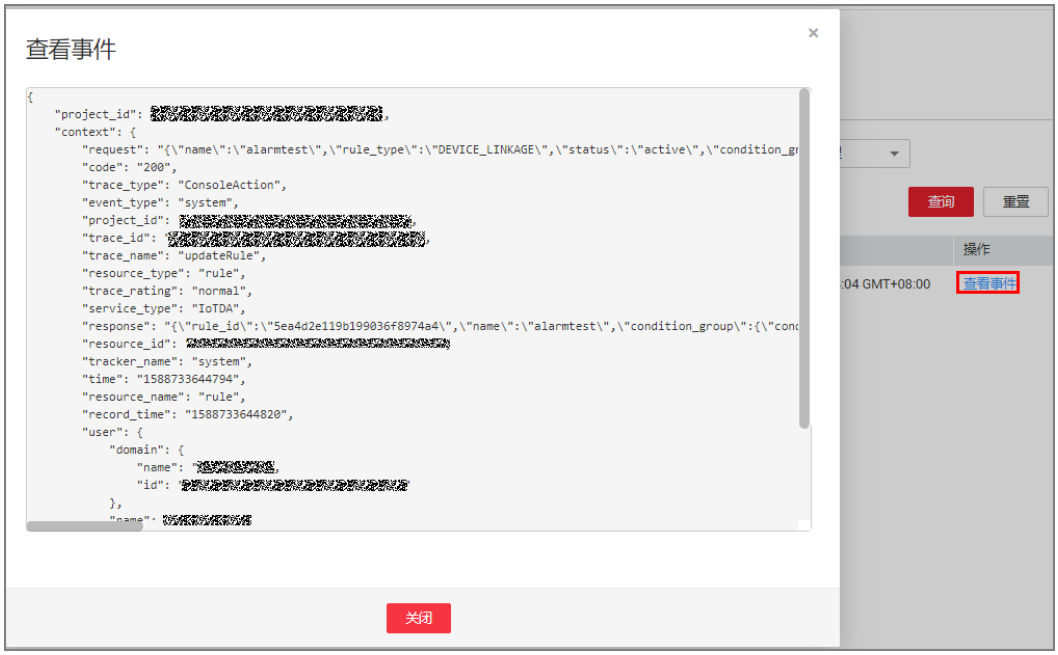
步骤2 单击左侧导航栏的“事件列表”，进入事件列表信息页面。

步骤3 设置对应的操作事件条件。当前事件列表支持四个维度的组合查询，详细信息如下：

- “事件类型”“事件来源”、“资源类型”、筛选类型”
  - “事件类型”：选择“管理事件”。
  - “事件来源”：选择“IoTDA”。
  - “资源类型”：选择“所有资源类型”，或者指定具体的资源类型。
  - “筛选类型”：选择“所有筛选类型”，或者选择“按资源ID”（选择该选项时，还需选择或者手动输入某个具体的资源ID）。
- 操作用户：在下拉框中选择某一具体的操作用户，此操作用户指用户级别，而非租户级别。
- 事件级别：可选项为“所有事件级别”、“normal”、“warning”、“incident”，只可选择其中一项。
- 时间范围：可选择查询最近七天内任意时间段的操作事件。

步骤4 单击“查询”，查看对应的操作事件。

步骤5 在需要查看的事件右侧，单击“查看事件”，弹出一个窗口，显示了该操作事件结构的详细信息。



关于云审计服务事件结构的关键字段详解，请参见《云审计服务用户指南》的**事件结构**章节。

----结束

6.5 查看运行日志

华为物联网平台支持记录平台与设备端，周边应用系统之间的对接情况，并以日志的形式上报到云日志服务（LTS），由LTS提供实时查询、海量存储、结构化处理和可视化图表分析能力，LTS每月免费赠送500M额度，超过后按需收费。LTS服务介绍和计费说明参见**云日志服务LTS**。

目前仅支持记录MQTT设备的业务运行日志，详细请参考下表。

业务类型	业务流程
设备状态	设备上线
	设备下线
设备消息	应用侧下发设备消息API
	物联网平台给设备下发消息
	设备给物联网平台上报消息
设备命令	应用侧下发设备命令API
	物联网平台给设备下发命令
设备属性	应用侧修改设备属性API
	设备属性上报
	网关批量上报设备属性
	平台设置设备属性

运行日志格式如下表，各字段之间以"|"分割。

字段	说明
时间	日志采集时间，时间格式为yyyy-MM-dd'T'HH:mm:ss,SSS'Z'， 例如：2020-06-16T09:24:45,708Z
设备Id	设备的deviceId
业务类型	对应业务类型： 设备状态：device.status 设备消息：device.message 设备命令：device.command 设备属性：device.property
操作	对应的操作名称，例如：API url，MQTT消息的Topic。
请求参数	操作的请求参数，例如：API的请求体。

字段	说明
结果信息	操作的结果，例如：API的响应体，错误信息等。
执行状态	操作状态码。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”，进入设备接入控制台。

步骤2 选择左侧导航栏的“监控运维 > 运行日志”，单击“配置日志采集开关”。

步骤3 在弹出的页面，勾选需要启动采集的业务类型，然后单击“确认”。

步骤4 创建运行日志转发规则，将采集的日志数据转发到其他云服务，以便您查看和处理，建议把日志数据转发至云日志服务LTS。以下以转发到云日志服务为例。

1. 选择左侧导航栏的“规则 > 数据转发”，单击“创建规则”。
2. 参考下表参数说明，填写规则内容，填写完成后单击“创建规则”。

参数名	参数说明
规则名称	创建的规则名称。
规则描述	对该规则的描述。

参数名	参数说明
数据来源	选择“运行日志”
触发事件	选择数据来源后，自动匹配触发事件。
资源空间	您可以选择单个资源空间或所有资源空间。

3. 单击“设置转发目标”页签，单击“添加”，设置转发目标。

参数名	参数说明
转发目标	选择“云日志服务（LTS）”
区域	当前仅支持转发至同区域的云日志服务。
日志组/日志流	选择LTS的日志组和日志流，未创建则访问**创建日志组**和**创建日志流**进行创建。

说明

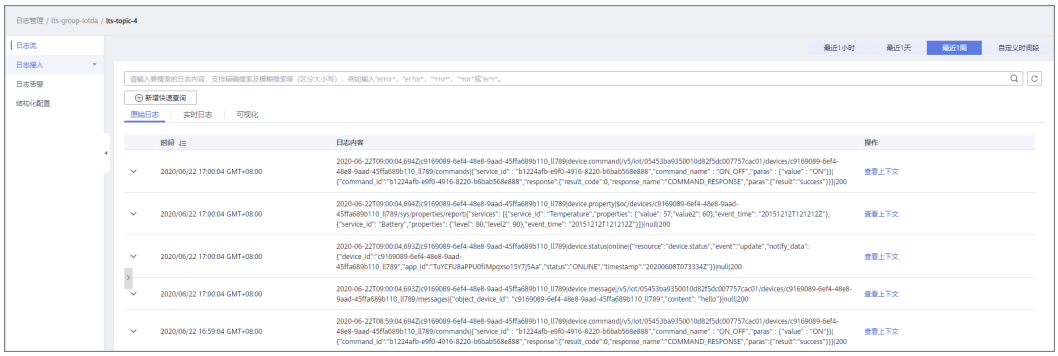
云日志服务创建的日志组默认存储时间为7天，超出存储时间的日志将被自动删除，您可以按需将日志数据转储至OBS桶中长期存储，具体步骤参考**日志转储至OBS**。

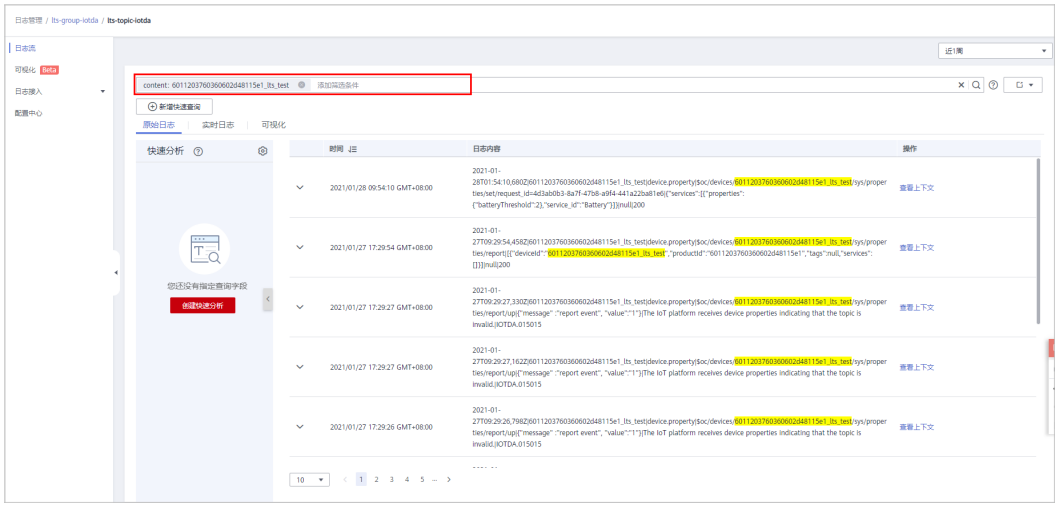
云日志服务每月免费赠送500M额度，默认超额继续采集，超过的部分按需收费，如果您需要关闭超额继续采集开关，登录云日志服务管理控制台，单击"配置中心"进行修改。

4. 单击“设置完成”，然后单击“启动规则”，实现运行日志转发至云日志服务。

步骤5 登录**云日志服务**管理控制台，单击“日志管理”。

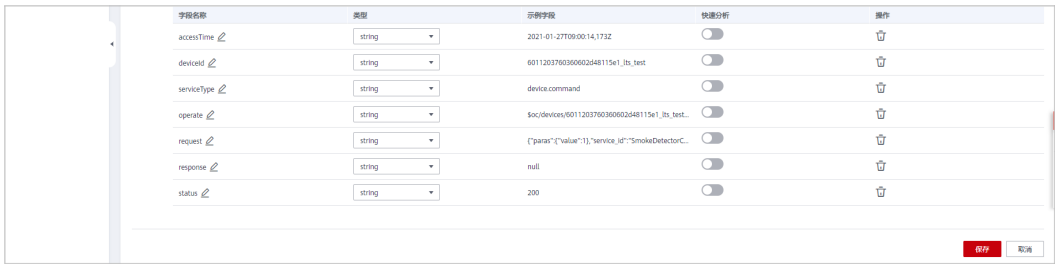
步骤6 选择**3**创建的日志组和日志流，查看IoTDA上报的日志信息。可参考**搜索日志**，对原始日志进行搜索，搜索目标日志。例如可以按照设备Id和业务类型搜索。



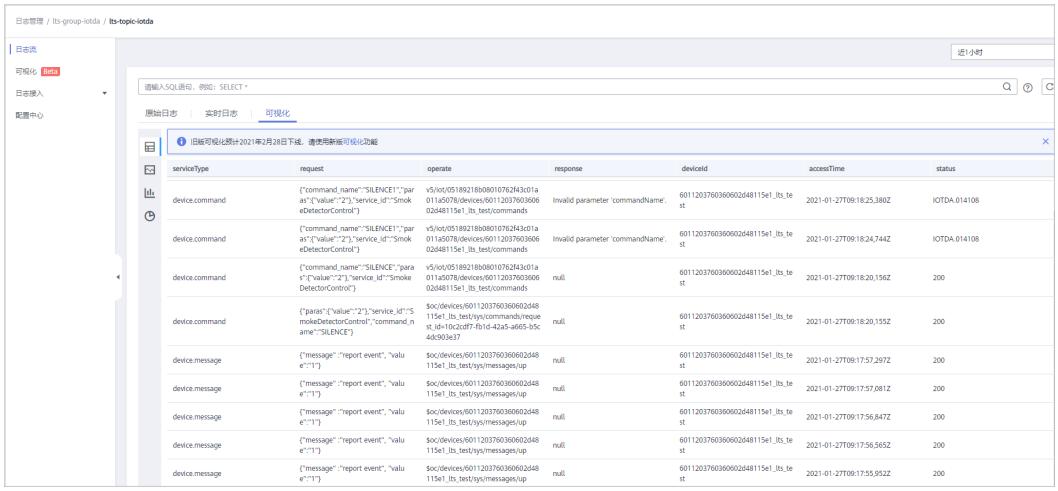


步骤7 在日志流详情页面，单击左侧导航栏的“可视化”，进入日志结构化配置页面，单击“分隔符”，推荐使用“分隔符 | ”作为日志提取方法。

步骤8 在选择示例日志页面，选择一条示例日志，日志结构依次为时间、设备ID、业务类型、操作、请求参数、结果信息、执行状态。选择指定分隔符"|",单击“智能提取”，修改智能提取的字段名称，如：accessTime、deviceId、serviceType、operate、request、response、status，修改字段类型为string。单击“保存”，完成日志的结构化配置，具体操作方法可参考**结构化日志**。



步骤9 对原始日志结构化后，触发响应业务。在“日志管理”页面，选择3创建的日志组和日志流，单击“可视化”，即可查看可视化之后的日志。可参考**SQL分析**对日志进行SQL查询。例如按照设备Id和业务类型查询：select * where deviceId = '5ebac4b54d9b0202c5d8ef0c_test_log' and serviceType = 'device.property'



----结束

7 存储管理

概述

物联网平台为用户提供数据存储服务，用于存储用户设备接入、设备日常管理时所产生的数据。目前设备接入控制台集成了IoT数据分析的存储管理服务，包括物联网温数据存储和物联网冷数据存储。

- 物联网温数据存储：针对物联网时序特征优化的存储，可大幅度提高数据压缩比，并且支持高效的时间维度聚合查询能力，推荐近期需要频繁查询的数据通过温存储方式保存。
- 物联网冷数据存储：采用价格低廉的对象存储方式，适合做物联网历史数据归档使用。通过IoT数据分析的离线分析引擎，可轻松实现PB级别的大数据分析。

存储配置

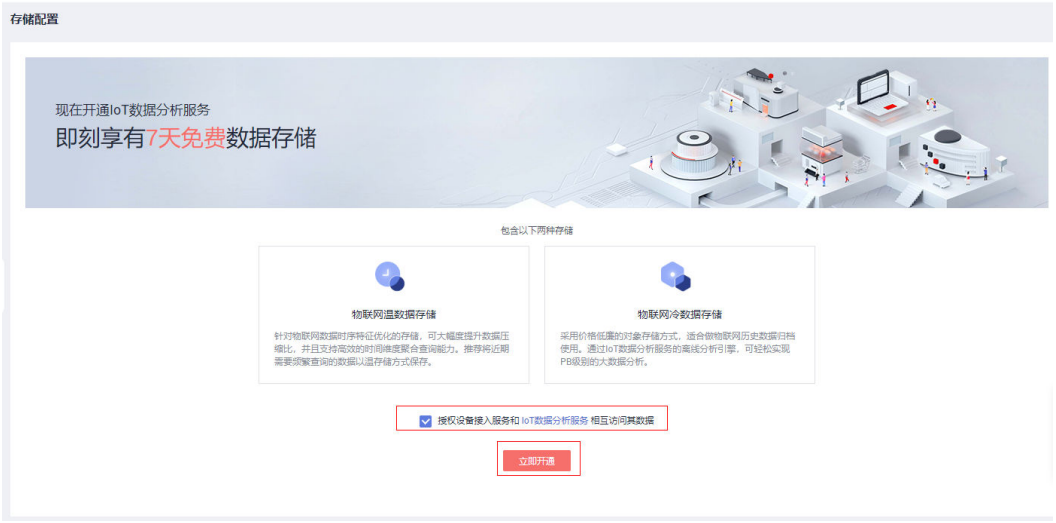
首次使用存储管理服务时，需要开通设备接入服务与IoT数据分析服务的相互访问数据功能。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

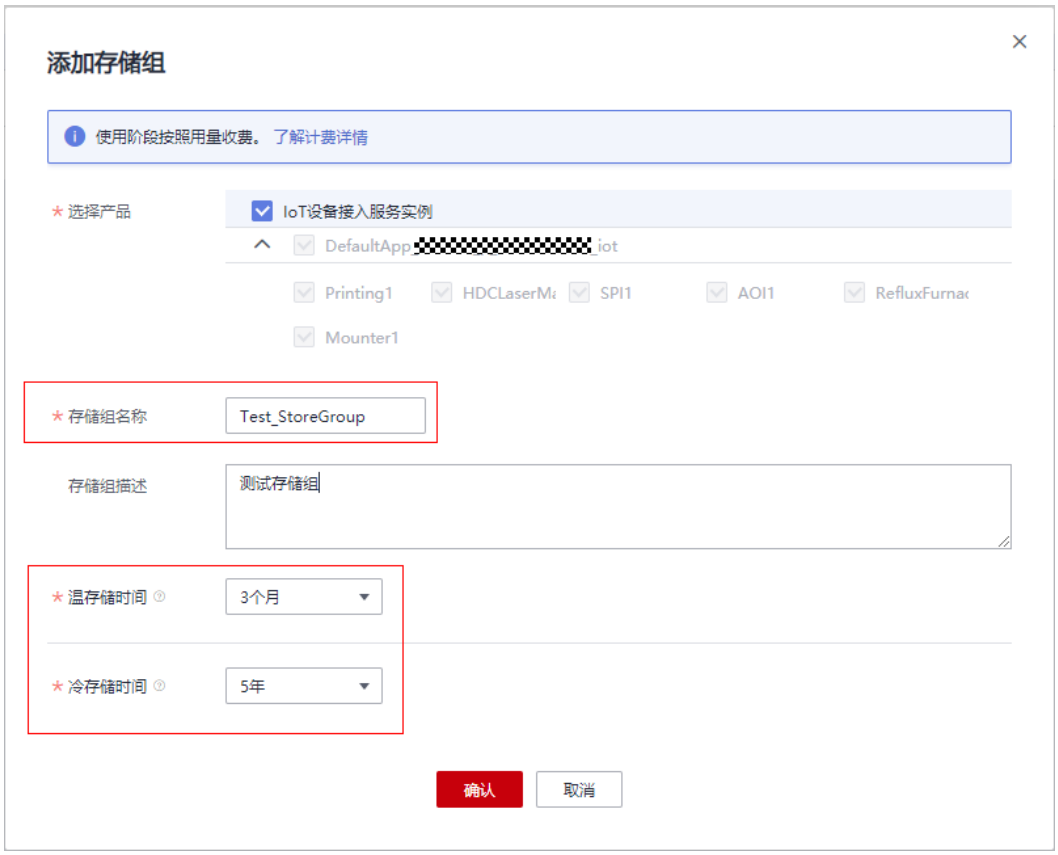
步骤2 在左侧导航栏，选择“存储管理”，勾选“授权设备接入和IoT数据分析服务相互访问其数据”，单击“**立即开通**”按钮。

说明

首次访问IoTA时会出现此授权提示， 授权成功后不会再次显示。

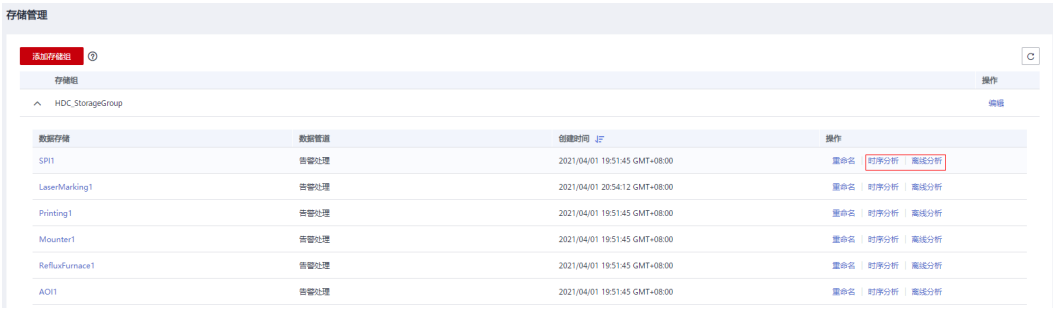


步骤3 单击左上方的“添加存储组”。在弹出的页面中选择需要使用存储服务的产品，填写存储名称，同时指定温数据和冷数据的存储时长，并单击“**确认**”。



添加存储组后，平台会自动为您在IoT数据分析服务中创建设备接入服务的数据源，并自动创建对应的存储与管道作业，将您选择的产品上报数据通过管道输出到对应存储中。

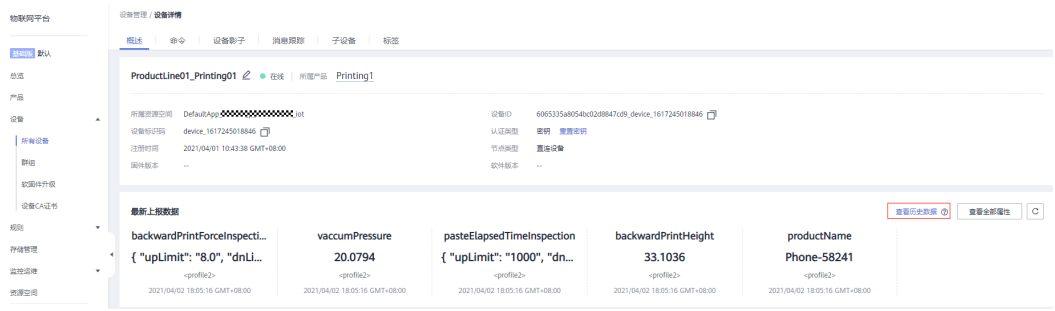
步骤4 展开创建的存储组，可以看到系统自动为您创建的存储，该存储保存了您对应产品上报的数据，单击“时序分析”或“离线分析”可进入IoTA数据分析服务进行更多操作。



说明

IoT数据分析服务：基于物联网资产模型，整合物联网数据集成、清洗、存储、分析、可视化，为开发者提供一站式服务，降低开发门槛，缩短开发周期，快速实现物联网数据价值变现。

完成存储配置后，您可以在设备列表页面，单击右上角的“分析历史数据”，批量分析和探索所有设备的历史数据。也支持在设备详情页面查看单个设备的历史数据。



----结束

8 资源空间

指在物联网平台中为您的业务划分的一个资源空间，您在平台中创建的资源（如产品、设备等）都需要归属到某个资源空间，您可以基于资源空间实现多业务应用的分域管理。

- 物联网平台允许用户最多创建10个资源空间，并默认首次开通服务时平台为用户自动创建的空间为默认资源空间。
- 创建资源空间时，物联网平台会分配一个app_id（接口调用时参数名为app_id）作为资源空间的唯一标识。
- 创建资源空间后，可以在资源空间中查看app_id。
- 默认资源空间不允许删除，其他资源空间删除后项目内的所有资源，如设备、产品、订阅数据在平台中的信息会被全部删除，并且不可恢复，请谨慎操作。

创建资源空间

用户首次开通设备接入服务时，物联网平台自动为用户创建了一个**默认资源空间**，“默认资源空间”每个实例仅有一个，不允许删除。

您可以基于默认资源空间创建产品，注册设备等，也可以参考如下步骤创建新的资源空间。

说明

如果您是老用户，即2020年04月27日00:00前开通的设备接入服务，平台设置默认资源空间的规则请参考**设置默认资源空间的规则**。

步骤1 访问**设备接入服务**，单击“管理控制台”进入设备接入控制台。

步骤2 在左侧导航栏选择“资源空间”，单击右上角的“新建资源空间”，在弹出的页面中，填写参数后，单击“确定”。

资源空间名称必须为账号下唯一。

----结束

查看资源空间

创建资源空间后，您可以在“资源空间列表”单击“详情”，查看该资源空间的APPID、创建时间、以及归属在该资源空间下的产品数、设备总数、群组数及创建的规则总数，若需要在其他资源空间下创建产品、设备、群组及规则，请在相应页面切换资源空间。

9 IoTDA 实例

9.1 实例介绍

华为物联网平台提供实例助力解决数据，资源隔离问题，当前华为云物联网平台提供基础版（共享实例）、标准版（标准实例）和企业版（专享实例）三种实例类型供您选择。

- **基础版**

  如果您开通设备接入服务后，物联网平台默认为您开通共享实例。在共享实例下，平台只提供基础的设备接入和设备管理功能，且多个租户共享统一的计算资源。如果您没有单独购买其他类型的实例，您创建的产品和设备均在共享实例中。

  说明：仅北京四提供基础版功能。

- **标准版**

  在标准实例下，平台提供更多的扩展功能，支持更多的服务组合能力以及行业解决方案集成适配，且不同租户的实例互相隔离。

  标准版提供免费试用，您可以在每日限定消息数和设备数范围内免费体验标准版的所有功能。

  如需更多设备数和消息数，请购买标准实例。购买标准实例后，免费试用下的所有资源自动转入到购买的新实例中。

  说明：免费试用对象仅限华为云账号实名认证为企业账号的用户。

- **企业版**

  购买专享实例后，专享实例运行在您指定的虚拟私有云内，您可以使用独立分配的公网和私网IP接入物联网平台。您购买的资源完全独享，资源安全性和隔离性更高。

关于您该选择哪种实例类型，请参考**产品规格说明**。

购买实例的约束说明

购买实例存在一些约束，当您登录后无法购买实例，或者购买实例后创建失败，请参考以下约束说明进行检查，并解除限制。

- 购买标准实例的约束说明

  - 同一个区域下，每个租户只能购买一个标准实例。
- 购买企业版实例的约束说明
  - 用户权限

    购买企业版实例前需要进行云服务访问授权操作，授权成功后，才可以购买企业版实例。详细操作请参考**配置云服务访问授权**。
  - 实例配额

    同一个VPC网络下最多只能购买5个企业版实例。
  - 子网中可用地址数量

    子网络中可用的地址数量不能少于10个。

前期规划

- 公私网对接方案

  公网对接：公网IP由平台提供，客户不需要规划。

  私网对接：在开通设备接入服务时需要指定VPC、子网等信息，默认设备接入服务只能从选定的子网接入，购买时需要提供合理规划的子网。如果使用设备接入服务的数据流转功能，比如数据转发到kafka，设备接入服务只能将数据转发到和它在同一个子网的kafka。如果您需要将数据转发至同一个VPC但不同子网的kafka服务，请提交工单说明您的诉求。
- VPC

  确定VPC的网段，当前VPC支持三个网段10.0.0.0/8~24、172.16.0.0/12~24和192.168.0.0/16~24。详细开通指南，请参考**创建虚拟私有云**。

  确定子网划分，子网的网段必须在VPC的网段范围内。
- 路由表

  确定路由表，默认路由表只能保证了同一个VPC下的所有子网互通。当默认路由表中的路由策略无法满足应用时，需要创建自定义的路由表。详细操作指南，请参考**路由表**。

配置云服务访问授权

华为物联网平台支持与华为云其它云服务进行对接。用户首次购买企业版实例时，需要进行访问授权操作。授权成功后可以创建端口、子网、安全组等信息，此外和终端节点服务对接，用于应用/设备通过私网接入物联网平台。

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 在左侧导航栏，选择“IoTDA实例 ”，单击企业版右侧的“服务授权 ”。

步骤3 在弹出的页面中单击“同意授权 ”，授权成功后设备接入服务将在**统一身份认证服务（IAM）**为您创建的名为iotda_admin_trust 的委托里添加了所需访问服务的权限。

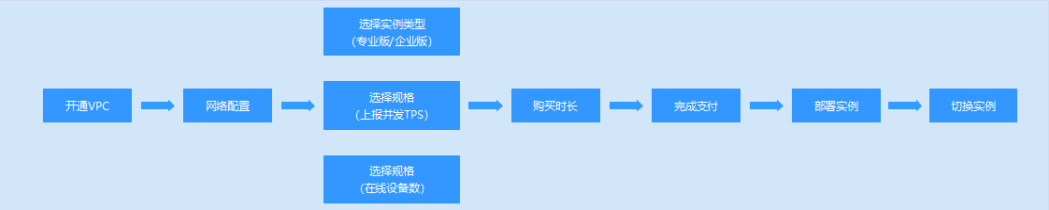
----结束

9.2 购买实例

购买企业版实例

购买企业版实例，可以享受更好的资源隔离和管控策略。购买的资源独享，具有更高的数据可靠性和安全性。

开通之前，请浏览整体开通流程，以便提高操作效率。



说明

每个租户最多只能购买15个企业版实例，如果需要更多实例，请**提交工单**说明您的诉求。

操作步骤

步骤1 配置访问授权后，在侧导航栏，选择“IoTDA实例”，单击企业版右侧的“购买实例”。

步骤2 在页面中填写实例配置信息，系统会根据您选择的“实例规格”和“购买时长”自动计算费用。

参数名称	参数说明
计费模式	选择实例的计费模式，当前仅支持“包年/包月”。
区域	设备接入服务部署的区域，当前支持“华北-北京四”、“华东-上海一”和“华南-广州-友好用户环境”。 **说明** 不同区域之间的云服务产品内网互不相通，请就近选择靠近您业务的区域，可降低网络时延，提高访问速度。
网络	选择虚拟私有云和子网。 如果需要创建新的虚拟私有云，请参考**虚拟私有云**创建虚拟私有云。
安全组	选择实例所关联的安全组，请参考**安全组**提前创建创建组。
公网接入	提供设备接入公网接入能力，请根据需要配置，避免不必要的浪费。
私网接入	• 勾选：购买实例时会自动购买VPC终端节点，并自动分配接入地址 • 未勾选：仍然需要私网接入，可以自行购买**VPC终端节点**对接。
接入端口	支持接入端口可配置，并提供了默认端口，支持可配置的端口如下： 应用接入：HTTPS(443)、AMQPS(5671) 设备接入：CoAP(5683)、CoAPS(5684)、MQTT(1883)、MQTTS(8883)、HTTPS(8943)
实例版本	企业版。
实例名称	设备接入实例的名字，方便根据名字管理实例。支持中文汉字、大小写字母、数字、下划线（_）和中划线（-），最大长度不超过64。目前实例名称创建好之后不支持修改。

参数名称	参数说明
实例描述	企业版实例的描述，可根据实例用户、实例的用途进行简单的描述。
购买时长	按月购买。 购买时可勾选自动续费，勾选后在实例到期时会自动对实例续费。

步骤3 单击“立即购买”，进入实例规格确认页面。

步骤4 规格确认无误后，单击“去支付”。

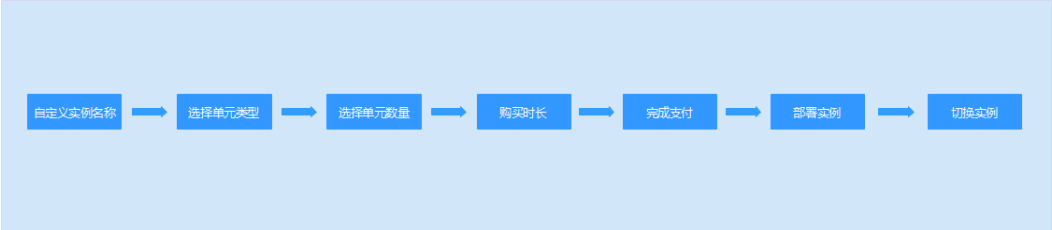
步骤5 在支付页面选择支付方式后，单击“确认付款”完成购买。

----结束

购买标准实例

标准实例提供灵活可配置的实例规格，适用于各类企业客户场景，可根据业务模型选购更经济的平台实例。

开通之前，请浏览整体开通流程，以便提高操作效率。



操作步骤

步骤1 在侧导航栏，选择“IoTDA实例”，单击标准版右侧的“购买实例”。

步骤2 在页面中填写实例配置信息，系统会根据您选择的“实例规格”和“购买时长”自动计算费用。

参数名称	参数说明
计费模式	选择实例的计费模式，当前仅支持“包年/包月”。
区域	设备接入服务部署的区域，当前支持“华北-北京四”、“华东-上海一”和“华南-广州-友好用户环境”。 **说明** 不同区域之间的云服务产品内网互不相通，请就近选择靠近您业务的区域，可降低网络时延，提高访问速度。

参数名称	参数说明
规格配置	支持中频单元S2和高频单元S3两种规格。 **说明** • IoTDA标准版实例的规格=实例包含的单元数量*每个单元的规格； • 一个标准版实例可以包含多个相同类型的单元，单个实例的单元数量上限为100个，单个实例的上下行消息TPS峰值最高到10万tps（比如实例包含100个S3，但峰值TPS最高只能到10万tps）； • 支持对实例的单元数进行在线升配和降配，如一个标准实例从3个S2变更为5个S2， 或者从3个S1减为1个S1。 • 同一个实例不支持混合叠加多个版本类型的单元，比如M个S2+N个S3。
实例名称	标准实例的名称，方便根据名称管理实例。支持中文汉字、大小写字母、数字、下划线（_）和中划线（-），最大长度不超过64。目前实例名称创建好之后不支持修改。
实例描述	标准实例的描述，可根据实例用户、实例的用途进行简单的描述。
购买时长	按月购买。购买时可勾选自动续费，勾选后在实例到期时会自动对实例续费。

步骤3 单击“立即购买”，进入实例规格确认页面。

步骤4 规格确认无误后，单击“去支付”。

步骤5 在支付页面选择支付方式后，单击“确认付款”完成购买。

----结束

9.3 实例管理

9.3.1 实例规格升配

IoTDA实例创建成功后，您可以根据业务需要，升级实例规格。

操作步骤

1. 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。
2. 选择左侧导航栏“IoTDA实例”，单击实例对应的“详情”进入实例详情页面。
3. 单击右上角的“升级配置”，根据您的业务需求，选择变更后的实例规格。

实例规格变更说明

实例类型	变更规格
标准版	服务单元规格、服务单元数量。
企业版	注册设备数上限。

9.3.2 切换实例类型

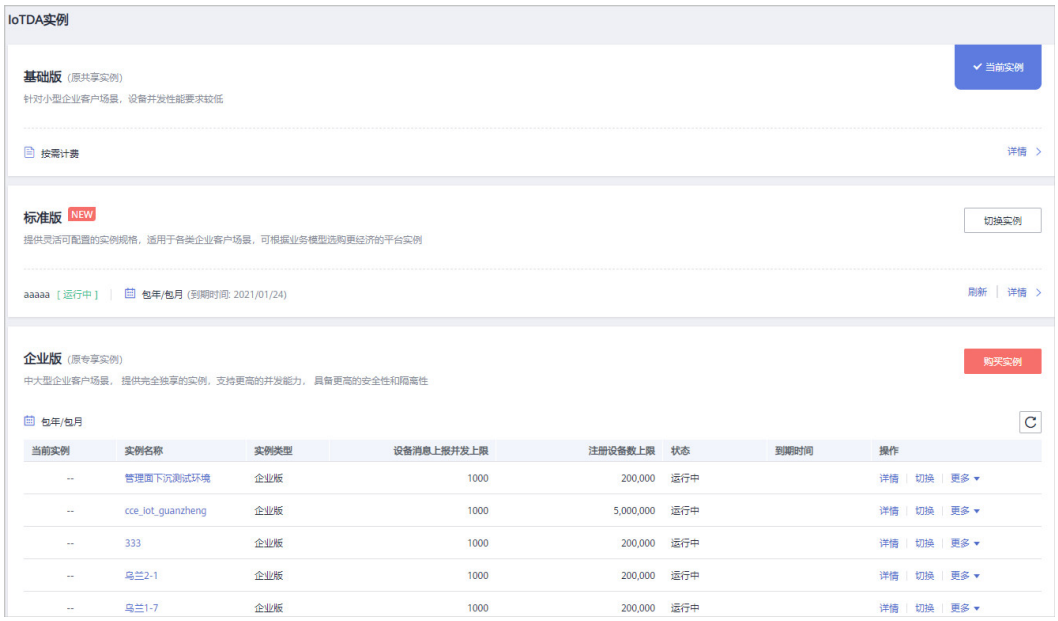
实例创建成功后，需要先切换到该实例，才可以在该实例下创建产品、设备和进行其他功能设置等。

在物联网平台左侧导航栏，会显示当前使用的实例类型。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 单击左侧导航栏“IoTDA实例”，会显示实例类型。您可以根据自己需求，单击“切换实例”切换当前实例类型。



----结束

9.3.3 实例续费/实例退订

当实例快到期需要续费继续使用的时候，您可以单击对应实例的“续费”按钮，进行续费操作。详细续费操作请参考**续费管理**。

针对不再继续使用的实例，您可以单击对应实例的“退订”按钮，释放您的云服务资源。详细退订操作请参考**退订管理**。

9.3.4 修改企业版实例信息

IoTDA企业版实例创建成功后，您可以根据自己的业务情况对IoTDA企业版实例的部分参数进行调整，包括实例名称、接入域名等信息。

操作步骤

步骤1 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。

步骤2 选择左侧导航栏“IoTDA实例”，单击企业版实例对应的“详情”进入实例详情页面。

步骤3 修改实例参数。

- 单击实例名称后的 ，修改实例名称。

  实例名称支持中文汉字、英文字母、数字、下划线（_）、中划线（-）。长度不能超过64个字符，一个中文汉字计一个字符。

- 您可以根据自身需要，单击接入域名下方的“未配置”，配置应用和设备接入平台的域名信息。

----结束

9.3.5 设置企业版实例公网访问

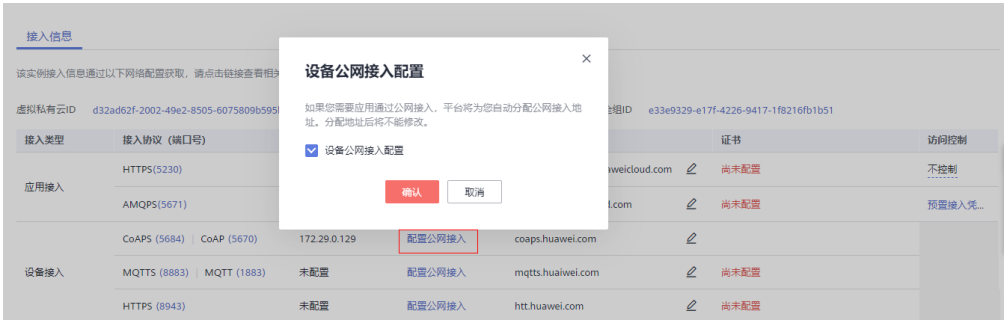
操作场景

为了助力企业快速迁移上云，物联网平台提供通过公网、私网、专线等接入形式。

当您需要通过公网地址访问设备接入实例时，可配置实例的公网访问功能。 公网访问带宽默认是5M/s，如果您需要扩大带宽，请**提交工单**说明您的需求。

操作步骤

1. 访问**设备接入服务**，单击“管理控制台 ”进入设备接入控制台。
2. 选择左侧导航栏“IoTDA实例”，单击实例对应的“详情”进入实例详情页面。
3. 单击“配置公网接入”，为应用或设备配置公网接入地址，单击“确认”后平台会自动生成公网IP。

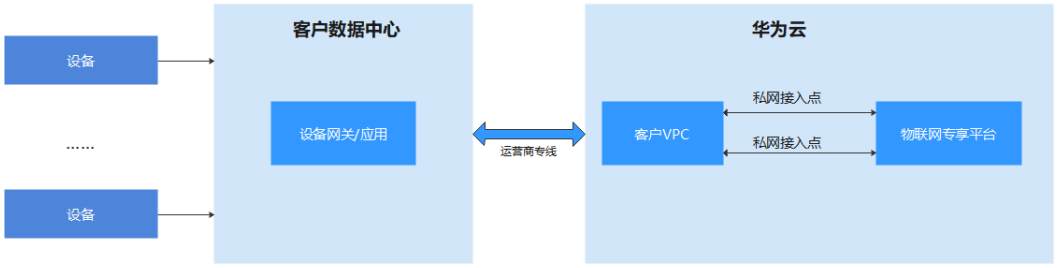


9.3.6 设置企业版实例专线访问

操作场景

如果用户已经有本地数据中心（设备接入网关、应用服务器等），可以通过云专线搭建本地数据中心与物联网企业版实例VPC之间专属连接通道，提供安全可靠的连接通道，助力用户快速迁移上云。

业务流程



1. 参考**云专线DC**搭建本地数据中心与华为云虚拟私有云通道。
2. 通过云专线访问物联网平台提供的私网接入地址。

10 权限管理

10.1 创建 IAM 用户并授权使用 IoTDA

如果您需要对您所拥有的IoTDA资源进行精细的权限管理，您可以使用**统一身份认证服务**（Identity and Access Management，简称IAM），通过IAM，您可以：

- 根据企业的业务组织，在您的华为云账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用IoTDA资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将IoTDA资源委托给更专业、高效的其他华为云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

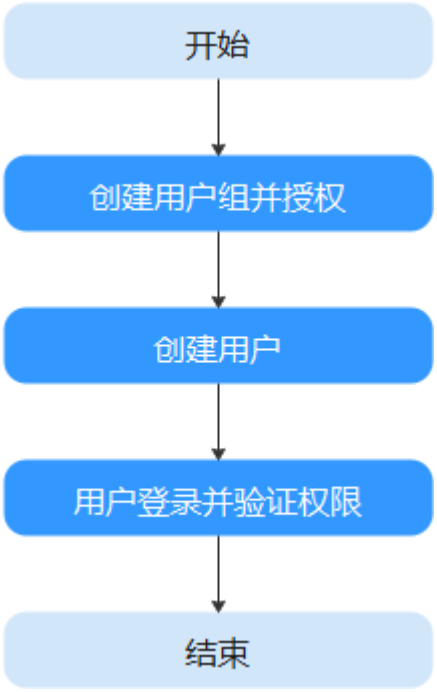
如果华为云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用IoTDA服务的其它功能。

本章节为您介绍对用户授权的方法。

前提条件

给用户组授权之前，请您了解用户组可以添加的IoTDA权限，并结合实际需求进行选择，请参考**表1 IoTDA系统权限**查看支持的系统权限。

示例流程



1. **创建用户组并授权**：在IAM控制台创建用户组，并授予用户组“IoTDA ReadOnlyAccess”权限。
2. **创建用户并加入用户组**：在IAM控制台创建用户，并将其加入**1**中创建的用户组。
3. **用户登录**并验证权限：新创建的用户登录控制台，按照以下操作验证权限是否生效。
   a. 参考**API列表**，根据文档示例调用应用侧API。
   b. 能调用查询设备接口，并且返回设备信息，但无法调用创建设备接口，同时返回‘Operation not allowed. The user does not have the permission’提示，表示“IoTDA ReadOnlyAccess”已生效。

   说明

   如果您需要此用户组拥有设备接入的所有权限，能够调用所有的API接口。可参考**这里**授予用户组IoTDA FullAccess预置系统策略权限。

如果系统策略不满足授权要求，管理员可以根据各服务支持的授权项，创建自定义策略，并通过给用户组授予自定义策略来进行精细的访问控制。自定义策略是对系统策略的扩展和补充，支持自定义操作和资源。目前华为云支持可视化视图、JSON视图两种自定策略配置方式。详细信息参考**创建自定义策略**。

10.2 创建 IoTDA 自定义策略

如果系统预置的权限，不满足您的授权要求，可以创建自定义策略。自定义策略中可以添加的授权项（Action）请参考 **表2 IoTDA操作与资源权限关系**。

目前华为云支持以下两种方式创建自定义策略：

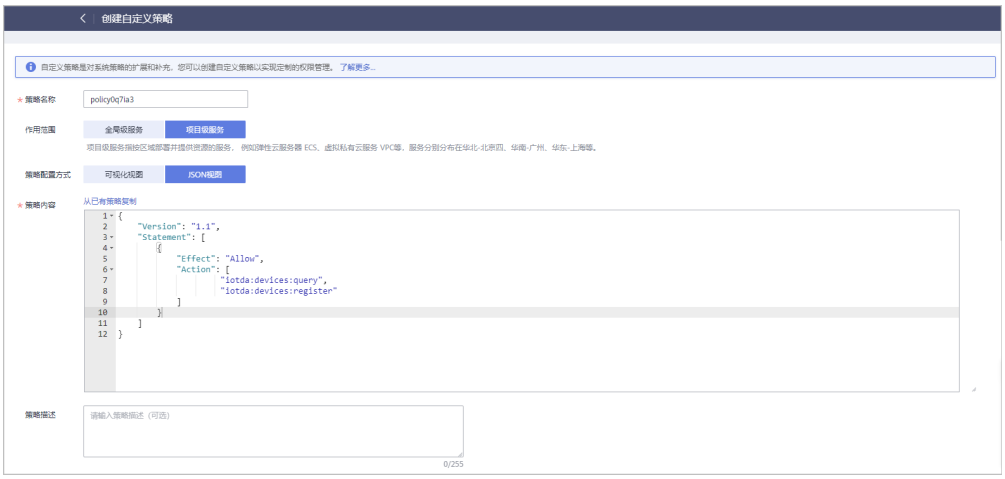
- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。

- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

具体创建步骤请参见：**创建自定义策略**。本章为您介绍常用的IoTDA自定义策略样例。

IoTDA 自定义策略样例

- **示例1：授权用户查询设备和注册设备**




```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
                  "iotda:devices:query",
                  "iotda:devices:register"
         ]
      }
   ]
}
```

- **示例2：拒绝用户删除设备**

拒绝策略需要同时配合其他策略使用，否则没有实际作用。用户被授予的策略中，一个授权项的作用如果同时存在Allow和Deny，则遵循Deny优先。

如果您给用户授予IoTDA FullAccess的系统策略，但不希望用户拥有IoTDA FullAccess中定义的删除设备权限，您可以创建一条拒绝删除设备的自定义策略，然后同时将IoTDA FullAccess和拒绝策略授予用户，根据Deny优先原则，则用户可以对IoTDA执行除了删除实例外的所有操作。拒绝策略示例如下：

```
{
      "Version": "1.1",
      "Statement": [
            {
                  "Effect": "Deny",
                  "Action": [
                        "iotda:devices:delete"
                  ]
            }
      ]
}
```

- **示例3：可视化视图创建自定义策略**

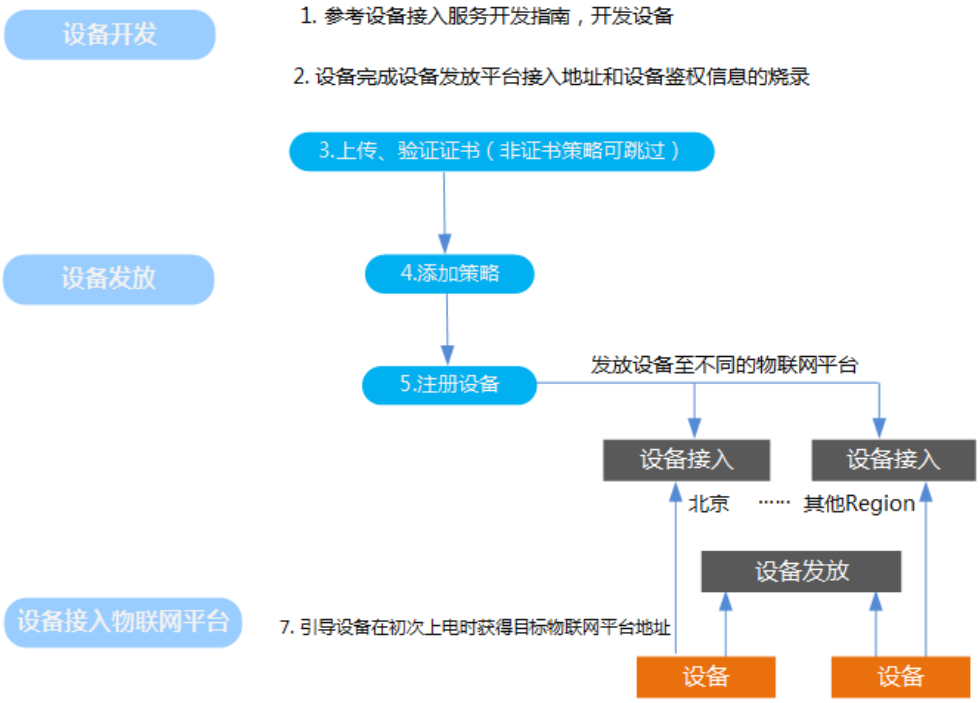
a. 访问**IAM控制台**，选择“权限”，单击右上角的“创建自定义策略”。

b. 参考下表填写参数，填写完成后，单击“确定”。

参数名	说明
策略名称	自定义
作用范围	选择“项目级服务”，表示系统权限中该服务的“所属区域”为“除全局区域外其他区域”。
策略配置方式	选择“可视化视图”。
策略内容	在策略内容下配置策略。 ● 选择“允许”。 ● “云服务”：选择设备接入（IoTDA）。 ● 操作：根据需求勾选产品权限. ● 选择资源：暂不支持 ● 请求条件：暂不支持
策略描述	（可选）输入策略描述。

11 设备发放

11.1 从这里开始



使用阶段	步骤
设备开发	1. 开通设备接入服务，参考开发指南，完成设备开发。 • LiteOS设备（LwM2M），设备需要具有设备发放功能。 • 原生MQTT协议设备，需要完成**设备引导接口**开发，使设备具有设备发放功能。 **注：**若设备需要使用物联网卡，可在**全球SIM联接**服务购买物联网卡和套餐。

使用阶段	步骤
	2. 设备完成设备发放平台接入地址和设备鉴权信息的烧录。**设备发放平台接入地址请在控制台中获取。** ● LiteOS设备（LwM2M）：需要烧录设备发放平台接入地址、设备标识码、引导服务端PSK。 ● 原生MQTT协议设备，需要烧录设备发放平台接入地址、设备ID、设备密钥。
设备发放	3. **证书**：用于证书策略发放的设备需要上传证书，防止通信数据在传输过程被篡改造成安全风险。
	4. **策略**：策略用于控制设备按照指定策略或规则发放至不同的物联网平台。
	5. **设备**：将设备基本信息导入设备发放平台中，用于后续发放至不同的物联网平台。
设备接入物联网平台	6. 设备初次上电时，先接入到设备发放平台，随后通过Bootstrap流程引导设备获得目标物联网平台地址。

11.2 证书

概述

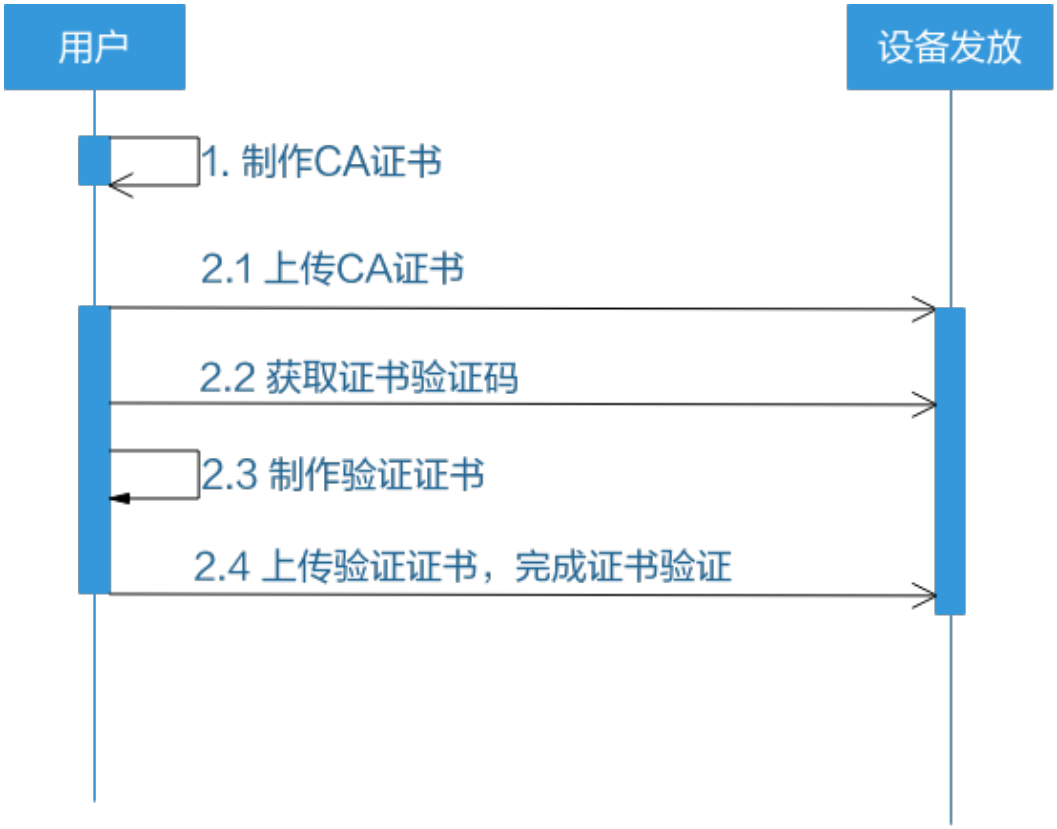
数字证书保证系统内各服务及系统与外部通信的安全性，防止通信数据在传输过程被篡改造成安全风险。

X.509是一种用于通信实体鉴别的数字证书，物联网平台支持设备使用自己的X.509证书进行认证鉴权。使用X.509认证技术时，设备无法被仿冒，避免了密钥被泄露的风险。

设备 CA 证书管理

本章节介绍设备CA证书的制作，以及如何在物联网平台上传、验证和管理设备CA证书。

图 11-1 设备 CA 证书制作



本章节样例中的各类证书常用文件名：

表 11-1

证书	文件名	MQTT.fx中的字段名
服务端证书	-	-
服务端CA证书	如下其中之一： huaweicloud-iot-root-ca-list.bks（android） huaweicloud-iot-root-ca-list.pem（c或java） huaweicloud-iot-root-ca-list.jks（java） bsca.jks（java） bsrootcert.pem（c）	CA File
设备证书（客户端证书）	deviceCert.crt	Client Certificate File
设备证书（客户端证书）私钥	deviceCert.key	Client Key File
CA证书（设备CA证书/客户端CA证书）	rootCA.crt	-

11.2.1 制作 CA 证书

本文以Windows环境为例，介绍通过Openssl工具制作CA证书和验证证书的方法。

⚠ 注意

以下“生成密钥对（rootCA.key）”和“生成CA证书（rootCA.crt）”为操作过程中需要使用到的两个文件。

制作 CA 证书

步骤1 在浏览器中访问这里，下载并进行安装OpenSSL工具，安装完成后配置环境变量。

步骤2 在 D:\certificates 文件夹下，以管理员身份运行cmd命令行窗口。

步骤3 生成密钥对（rootCA.key）：

说明

生成“密钥对”时输入的密码在生成“证书签名请求文件”、“CA证书”，“验证证书”以及“设备证书”时需要用到，请妥善保存。

```
openssl genrsa -des3 -out rootCA.key 2048
```

步骤4 使用密钥对生成证书签名请求文件：

说明

生成证书签名请求文件时，要求填写证书唯一标识名称（Distinguished Name，DN）信息，参数说明如下表1 所示。

表 11-2

提示	参数名称	取值样例
Country Name (2 letter code) []:	国家/地区	CN
State or Province Name (full name) []:	省/市	GuangDong
Locality Name (eg, city) []:	城市	ShenZhen
Organization Name (eg, company) []:	组织机构（或公司名）	Huawei Technologies Co., Ltd.
Organizational Unit Name (eg, section) []:	机构部门	Cloud Dept.
Common Name (eg, fully qualified host name) []:	CA名称（CN）	Huawei IoTDP CA
Email Address []:	邮箱地址	/
A challenge password []:	证书密码，如您不设置密码，可以直接回车	/

提示	参数名称	取值样例
An optional company name []:	可选公司名称，如您不设置，可以直接回车	/

```
openssl req -new -key rootCA.key -out rootCA.csr
```

步骤5 生成CA证书（rootCA.crt）：

```
openssl x509 -req -days 50000 -in rootCA.csr -signkey rootCA.key -out rootCA.crt
```

说明

“-days”后的参数值指定了该证书的有效天数，此处示例为50000天，您可根据实际业务场景和需要进行调整。

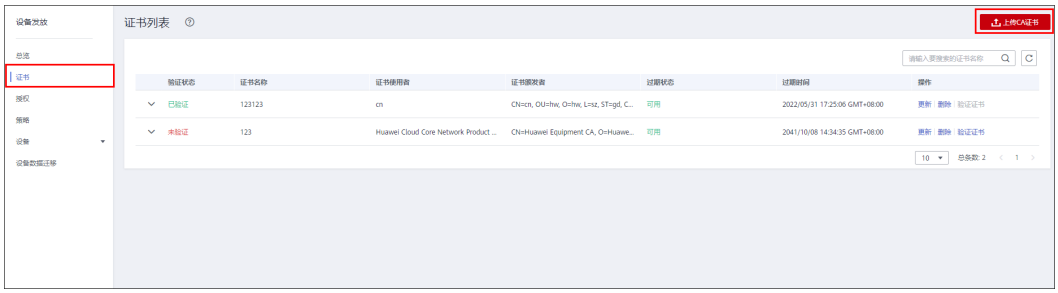
----结束

11.2.2 上传 CA 证书

操作步骤

步骤1 登录**设备发放控制台**。

步骤2 在设备发放控制台，左侧导航窗格中，选择“证书”，单击右上方的“上传CA证书”。



步骤3 在“上传CA证书”页面，填写“证书名称”，单击“添加文件”，上传此前“**制作CA证书**”步骤中生成的“CA证书（rootCA.crt文件）”，单击“确定”。

----结束

说明

上传的CA证书初始状态为“未验证”，需要完成“**验证CA证书**”过程，方可正常使用该CA证书。

证书状态表

CA证书状态	说明
已验证	可正常使用。
未验证	不可正常使用，待验证通过后，方可正常使用。
已过期	CA证书已过期，需更新，但不影响平台使用该CA证书验证对应的设备证书。
即将过期	CA证书30天内即将过期，需及时更新。

11.2.3 验证 CA 证书

对于已上传的CA证书，平台要求用户完成“验证CA证书”过程，以验证用户具备该CA证书的签发能力。

操作步骤

步骤1 登录**设备发放控制台**。

步骤2 在设备发放控制台，左侧导航窗格中，选择“证书”，单击“证书列表”条目的操作栏中的“验证证书”。

步骤3 在上传验证证书页面，单击“生成验证码”，单击“复制图标”复制此CA证书的随机验证码。

说明

CA证书验证码有效期为一天，请及时使用验证码生成验证证书并完成验证。

验证码的生成为替换机制，即对于一个CA证书，即使此前的验证码未过期，也将被新生成的验证码替换。

步骤4 使用OpenSSL工具为验证证书生成密钥对。

```
openssl genrsa -out verificationCert.key 2048
```

步骤5 利用此验证码生成证书签名请求文件CSR。

```
openssl req -new -key verificationCert.key -out verificationCert.csr
```

说明

CSR文件的Common Name (e.g. server FQDN or YOUR name) 需要填写此验证码。

步骤6 使用CA证书、CA证书私钥和上一步骤中生成的CSR文件创建验证证书（verificationCert.crt）。

```
openssl x509 -req -in verificationCert.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out verificationCert.crt -days 36500 -sha256
```

说明

生成验证证书用到的“rootCA.crt”和“rootCA.key”这两个文件，为“**制作CA证书**”中所生成的两个文件。

“-days”后的参数值指定了该证书的有效天数，此处示例为36500天，您可根据实际业务场景和需要进行调整。

步骤7 上传验证证书进行验证。

----结束

11.2.4 更新 CA 证书

步骤1 登录**设备发放控制台**。

步骤2 在设备发放控制台，左侧导航窗格中，选择“证书”，单击“证书列表”条目的操作栏中的“更新”。

说明

更新CA证书前，要求该证书未被设备、策略、注册组关联。

更新CA证书后，该证书状态将变为未验证，请重新完成验证CA证书过程。

----结束

11.2.5 删除 CA 证书

步骤1 登录设备发放控制台。

步骤2 在设备发放控制台，左侧导航窗格中，选择“证书”，单击“证书列表”条目的操作栏中的“删除”。

说明

关联了至少一个设备、策略或注册组的CA证书，不允许删除。

请谨慎操作，删除后的CA证书的所有数据将被删除且不可恢复。

----结束

11.3 策略

策略用于控制设备按照指定策略或规则发放至不同的物联网平台。当前支持自定义策略、证书策略、静态策略。

11.3.1 自定义策略

与静态策略和证书策略相比，自定义策略为您提供更灵活的策略机制。自定义策略实例指示函数关联的设备接入实例，一个函数可关联多个设备接入实例。

设备匹配自定义策略实例的机制为：

1. 设备的发放策略指定为“函数策略”时，需同时指定其关联的函数，设备与自定义策略实例关联同一个函数时，即被认为该设备匹配上该条策略实例；
2. 一个设备可匹配多条自定义策略实例。

11.3.1.1 创建自定义策略函数

了解函数接口定义

函数服务对函数有明确的接口定义。

说明

以java语言为例，接口定义为：*作用域 返回参数 函数名（函数参数，Context参数）*。

- 作用域：提供给FunctionGraph调用的用户函数必须定义为public。
- 返回参数：用户定义，FunctionGraph负责转换为字符串，作为HTTP Response返回。对于返回参数对象类型，HTTP Response该类型的JSON字符串。
- 函数名：用户定义函数名称。
- 用户定义参数：当前函数只支持一个用户参数。对于复杂参数，建议定义为对象类型，以JSON字符串提供数据。FunctionGraph调用函数时，解析JSON为对象。
- Context参数：runtime提供函数执行上下文，其接口定义在**SDK接口**说明。

创建Java函数时，函数入口参数需要提供函数完整的名字空间，参数格式为：包名.类名.函数名。

设备发放在此基础上，要求函数代码满足如下条件：

- 返回参数：需满足**设备发放对返回参数的约束**；
- 函数参数：需满足**设备发放对函数参数的约束**；
- 函数接口实现：从函数参数中的备选接入点中选择一个接入点，调用**发放设备接口**，根据接口响应拼接参数返回。

编写自定义函数

步骤1 创建函数工程，编写函数。

首先建立一个普通的Java项目，添加FunctionGraph函数**JavaSDK**为工程依赖，可下载设备发放提供**函数Demo**，参照Demo创建工程编写函数。

说明

FunctionGraph函数JavaSDK提供了Event事件接口、Context接口和日志记录接口。其中Runtime-x.x.x.jar包含了函数执行上下文，其他Jar包为附带的第三方库，可按需添加。

Java函数开发指南参见**函数工作流 FunctionGraph > 开发指南 > 如何开发函数 > Java函数开发指南**。

参照如下代码编写函数，实现必要逻辑。

```
package com.huawei.demo;

import com.huawei.demo.common.logger.DefaultLogger;
import com.huawei.demo.model.AccessPointPara;
import com.huawei.demo.model.FunctionGraphPara;
import com.huawei.demo.model.TdpFuncResult;
import com.huawei.services.runtime.Context;

import java.util.Optional;

/**
 * 实现该类。
 * WARNING：{@link #apiHandle(FunctionGraphPara, Context)} 该方法必须在子类中定义，否则函数无法触发该函数接口！！！
 * <pre>
 * {@code
 *     @Override
 *     public TdpFuncResult apiHandle(FunctionGraphPara para, Context context) {
 *         return super.apiHandle(para, context);
 *     }
 * }
 * </pre>
 */
```

```
public abstract class TdpFunction {
    protected static final DefaultLogger LOGGER = new DefaultLogger(TdpFunction.class);

    /**
     * 函数定义
     */
    public TdpFuncResult apiHandle(FunctionGraphPara para, Context context) {
        // 获取日志
        DefaultLogger.init(Optional.ofNullable(context)
            .map(Context::getLogger)
            .orElse(null));

        // 校验入参
        TdpFuncResult result = checkPara(para, context);
        if (result != null) {
            return result;
        }

        // 确定接入点
        AccessPointPara accessPointPara = determineAccessPoint(para);

        // 发放设备
        result = provisionDevice(para, accessPointPara);

        LOGGER.info("result:{}", result);
        return result;
    }

    /**
     * 校验入参
     */
    protected abstract TdpFuncResult checkPara(FunctionGraphPara para, Context context);

    /**
     * 从备选的接入点中选择合适的接入点
     */
    protected abstract AccessPointPara determineAccessPoint(FunctionGraphPara para);

    /**
     * 调用设备发放的发放接口发放设备
     */
    protected abstract TdpFuncResult provisionDevice(FunctionGraphPara para, AccessPointPara
accessPointPara);
}
```

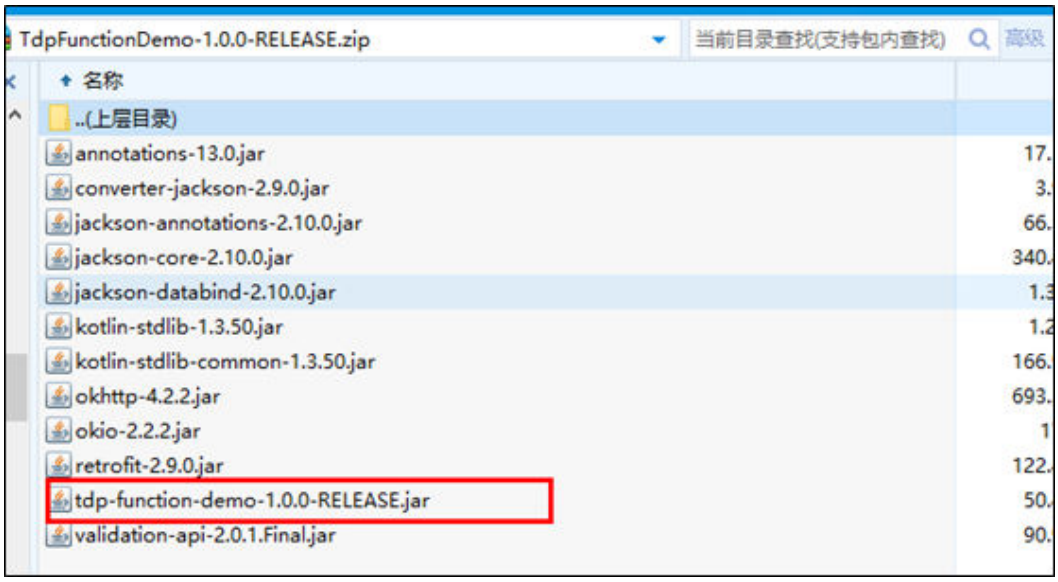
步骤2 打包函数，上传代码。

说明

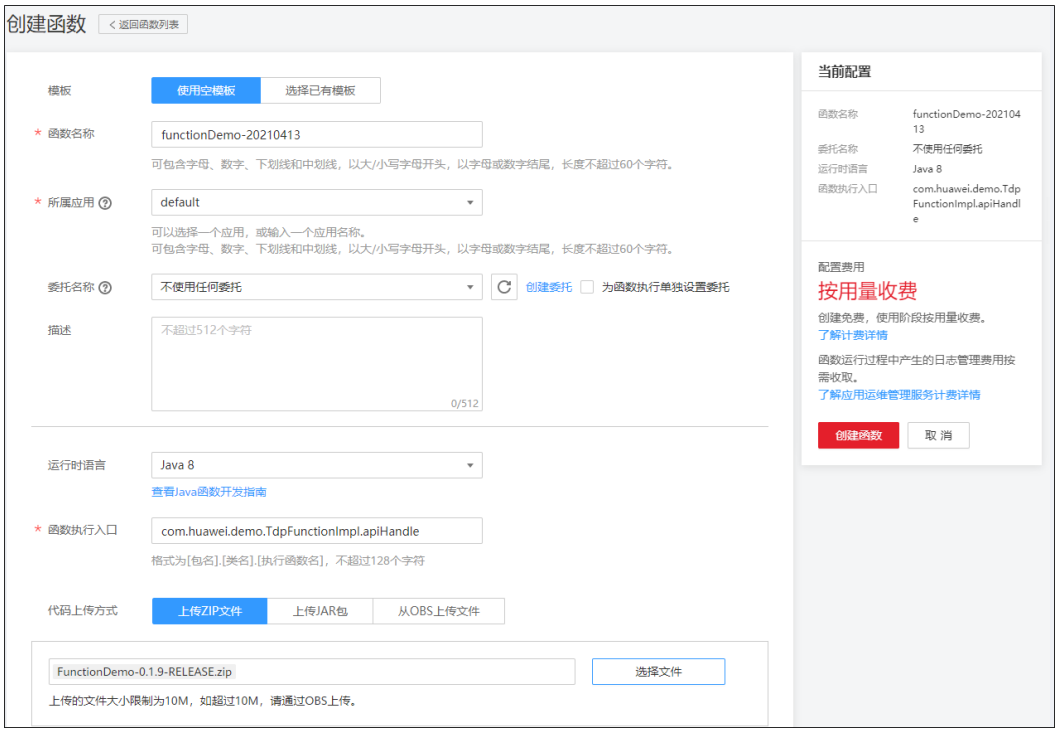
函数打包规范参见**函数工作流 FunctionGraph > 开发指南 > 概述 > 函数工程打包规范**。

如函数工程未使用到第三方库，则可将工程打成一个Jar包上传。如使用到第三方库，则需将工程Jar包和第三方Jar包打包成一个Zip包。

本文使用到了多个第三方库，因此，如下图所示，将工程Jar包和第三方Jar包打包成一个Zip包。



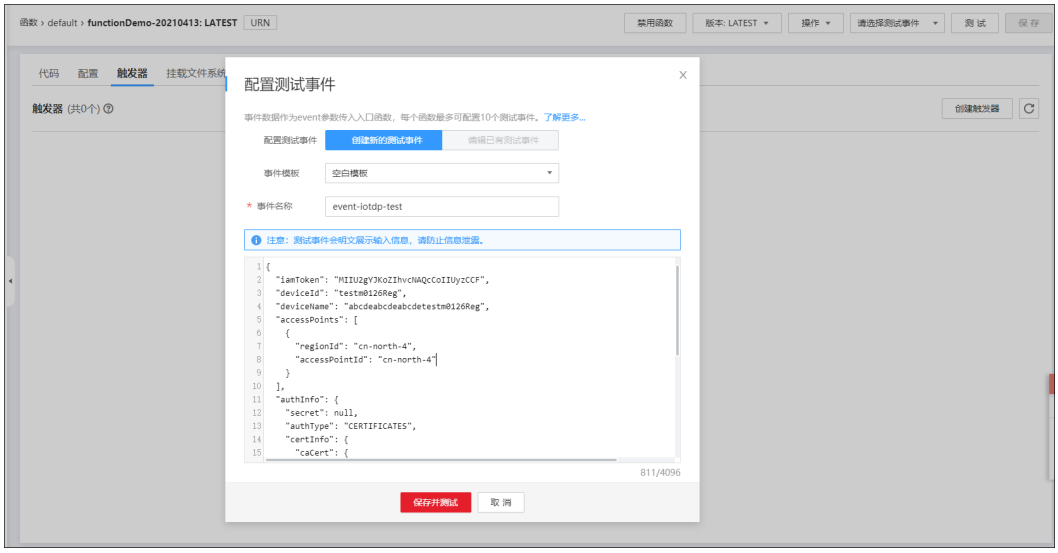
进入函数工作流服务控制台，创建函数并上传包含工程Jar包和第三方Jar包的Zip包。



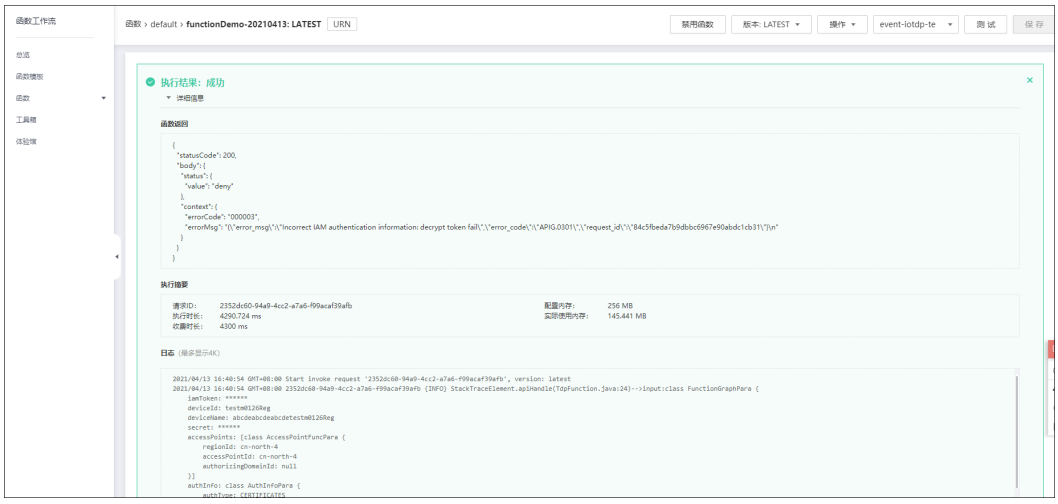
为避免调试过程中函数因内存不足或超时导致调用失败，视实际使用情况调整内存大小和超时时间。

步骤3 配置测试事件，调试函数。

创建测试事件，填写满足设备发放约束的函数参数的JSON格式。



点击“保存并测试”，查看函数运行结果，确认函数逻辑的正确性。



日志显示，已调用了发放设备接口，函数主要逻辑正确。至此，您已完成自定义策略函数的编写，可进行后续步骤。

----结束

后续操作

完成自定义策略函数的编写后，创建关联该函数的自定义策略实例以及设备或注册组，通过设备南向操作触发函数，将设备发放到指定目的接入点（设备接入实例）。

后续操作包括：

- **添加自定义策略实例**；
- **注册设备**或**创建注册组**；
- 南向接口：**MQTT CONNECT连接鉴权**、**设备请求引导消息**、**设备接收引导消息**。

最佳实践：

- 结合函数服务通过自定义策略发放证书认证的设备。

设备发放对返回参数的约束

表 11-3 TdpFuncResult

名称	说明	类型
statusCode	String	函数执行状态码，用于标识函数执行过程是否出现异常，遵循HTTP状态码含义。
body	String	字符串，但格式为JSON，结构为TdpFuncBody。

表 11-4 TdpFuncBody

名称	说明	类型
status	String	设备发放业务功能标识，allow表示发放成功，deny表示发放失败，设备发放使用此标识判断函数内业务功能执行成功与否。
context	TdpFuncBodyContext	扩展字段，用于承载函数执行结果。

表 11-5 TdpFuncBodyContext

名称	说明	类型
allocationResult	String	发放结果，存放发放接口返回的响应结构体。
errorCode	String	错误码，如发放失败，则此值需不为空。
errorMsg	String	错误描述，如发放失败，则此值需不为空。

发放设备成功的返回参数样例

```
{
  "statusCode": 200,
  "body": "{\"status\":\"allow\",\"context\":{\"allocationResult\":\"dps返回的下发结果\"}}"
}
```

发放设备失败的返回参数样例

```
{
  "statusCode": 200,
  "body": "{\"status\":\"deny\",\"context\":{\"errorCode\":\"错误码\",\"errorMsg\":\"错误描述\"}}"
}
```

设备发放对函数参数的约束

表 11-6 FunctionGraphPara

名称	类型	说明
iamToken	String	委托Token，便于用户调用发放接口。
deviceId	String	设备ID。
deviceName	String	设备名称。
accessPoints	List<AccessPointPara>	备选接入点列表。
authInfo	AuthInfoPara	设备认证信息。

表 11-7 AccessPointPara

名称	类型	说明
regionId	String	备选接入点的区域ID。
accessPointId	String	备选接入点的接入点ID。
authorizingDomainId	String	备选接入点的接入点授权租户，仅在授权场景下使用。

表 11-8 AuthInfoPara

名称	类型	说明
authType	String	认证类型。可选 SECRET 和 CERTIFICATES。
secret	String	设备密钥信息，当认证类型为 SECRET 时携带。
certInfo	CertInfoPara	证书信息，当认证类型为 CERTIFICATES 时携带。

表 11-9 CertInfoPara

名称	类型	说明
caCert	CaCertPara	设备CA证书信息。
deviceCert	List<DeviceCertPara>	设备证书信息。

表 11-10 CaCertPara

名称	类型	说明
subjectCnName	String	证书使用者CN NAME。
caCertName	String	证书名。
fingerprint	String	证书指纹。

表 11-11 DeviceCertPara

名称	类型	说明
subjectCnName	String	证书使用者CN NAME。
issuerCnName	String	证书颁发者CN NAME。
sha1Fingerprint	String	证书指纹（使用SHA-1算法计算）。
sha256Fingerprint	String	证书指纹（使用SHA-256算法计算）。

证书认证设备触发函数策略的函数参数样例：

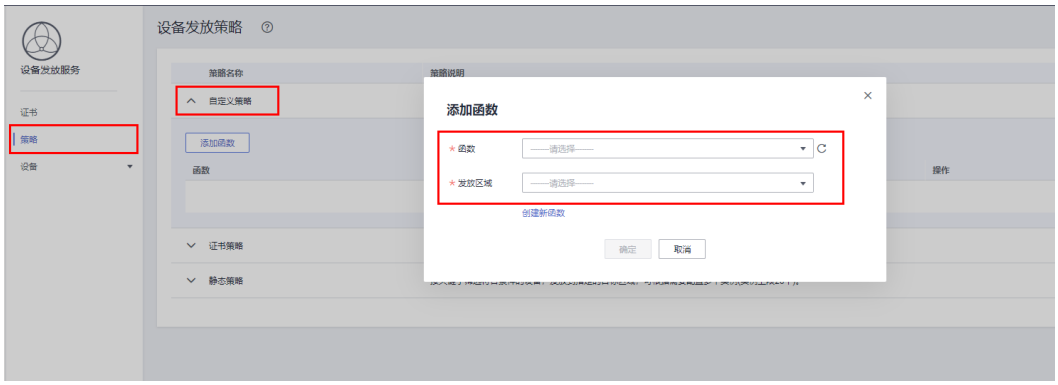
```
{
  "iamToken": "MIIT3gYJKoZIhvcNAQcCoIIT...",
  "deviceId": "testDeviceId",
  "deviceName": "testDeviceName",
  "accessPoints": [
    {
      "regionId": "cn-north-4",
      "accessPointId": "cn-north-4",
      "authorizingDomainId": null
    },
    {
      "regionId": "cn-north-4",
      "accessPointId": "86a45e59-3003-24b8-4e81-df0a9a694639",
      "authorizingDomainId": "84fd2c95ddf03b840f18100a000d45c2"
    }
  ],
  "authInfo": {
    "authType": "CERTIFICATES",
    "certInfo": {
      "caCert": {
        "subjectCnName": "serverCommonName20200922204814",
        "caCertName": "ca20200922204814",
        "fingerprint": "dc0f1016f495157344ac5f1296335cff725ef22f"
      },
      "deviceCert": [
        {
          "subjectCnName": "deviceCommonName20200922204814",
          "issuerCnName": "serverCommonName20200922204814",
          "sha1Fingerprint": "b2217bd882968b0bc15b1c2b132ba8a598f11879",
          "sha256Fingerprint": "734326bb5f3065d8e90d95ce1910b831d3856a1318770768bf9b03923d641ddd"
        }
      ]
    }
  }
}
```

密钥认证设备触发函数策略的函数参数样例：

```
{
  "iamToken": "MIIT3gYJKoZIhvcNAQcCoIIT...",
  "deviceId": "testDeviceId",
  "deviceName": "testDeviceName",
  "accessPoints": [
    {
      "regionId": "cn-north-4",
      "accessPointId": "cn-north-4",
      "authorizingDomainId": null
    },
    {
      "regionId": "cn-north-4",
      "accessPointId": "86a45e59-3003-24b8-4e81-df0a9a694639",
      "authorizingDomainId": "84fd2c95ddf03b840f18100a000d45c2"
    }
  ],
  "authInfo": {
    "authType": "SECRET",
    "secret": "29ccf3ff01247d731fef"
  }
}
```

11.3.1.2 添加自定义策略实例

步骤1 进入“策略”界面，单击展开“自定义策略”，单击“添加函数”。



步骤2 按照下方参数说明填写关键参数信息后，单击“确定”。

参数名称	说明	示例
函数	即在**函数服务**中实现的自定义策略。如果下拉框没有你想要的函数，可以单击创建新函数来实现你的自定义策略需求。	将需要通过函数“function”发放的设备发放至华北-北京四的物联网平台。 ● 需通过函数“function”发放的设备：WaterMeter-Beijing0001、WaterMeter-Beijing0002 ● 函数：function ● 发放区域：华北-北京四
发放区域	发放到指定区域后，设备将接入对应区域的设备接入服务。 所选区域未开通设备接入服务时，如果确定添加实例，系统将自动为您开通设备接入服务。不同区域设备接入服务价格不同，收费详情请参考价格说明。	

----结束

11.3.2 证书策略

概述

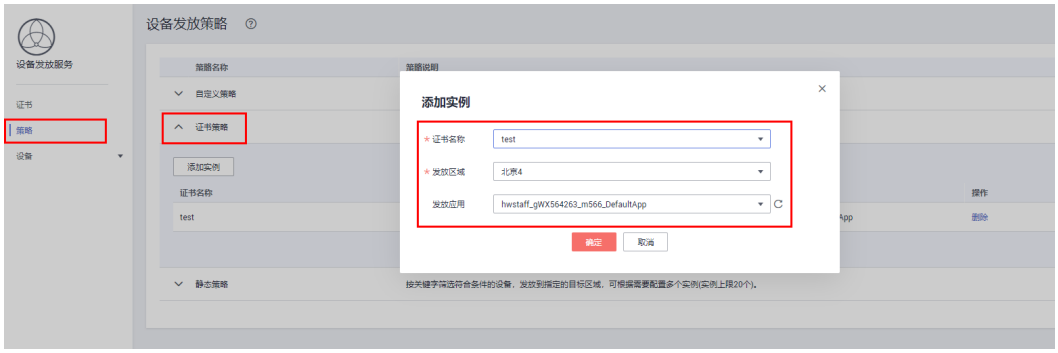
证书策略，即通过平台认证设备的设备CA证书匹配的发放策略。每条证书策略实例指示匹配上该策略实例的设备将会被发放到该策略实例关联的设备接入实例的对应资源空间（即应用）下。

设备匹配证书策略实例的机制为：

1. 设备的发放策略指定为“证书策略”时，其认证方式也必须为X.509证书认证且同时指定了认证设备的设备CA证书，当设备关联的设备CA证书与证书策略实例关联的证书为同一个证书时，即被认为该设备匹配上该条策略实例；
2. 一个设备最多匹配一条证书策略实例；一个CA证书最多被一条证书策略关联。

操作步骤

步骤1 进入“策略”界面，单击展开“证书策略”，单击“添加实例”。



步骤2 按照下方参数说明填写关键参数信息后，单击“确定”。

参数名称	说明	示例
证书名称	即所要根据证书属性将设备发放到指定的目标区域，选择对应的证书。	将需要通过证书“certificates”发放的设备发放至华北-北京四的物联网平台。 • 需通过证书“certificates”发放的设备：WaterMeter-Beijing0001、WaterMeter-Beijing0002 • 证书名称：certificates • 发放区域：华北-北京四 • 发放应用：beijing-app1
发放区域	发放到指定区域后，设备将接入对应区域的设备接入服务。 所选区域未开通设备接入服务时，如果确定添加实例，系统将自动为您开通设备接入服务。不同区域设备接入服务价格不同，收费详情请参考价格说明。	
发放应用	选择对应设备接入服务区域已创建的应用。在物联网平台中，设备由应用统一管理。 如果对应设备接入服务区域未创建应用，需要前往对应服务创建应用。	

----结束

11.3.3 静态策略

概述

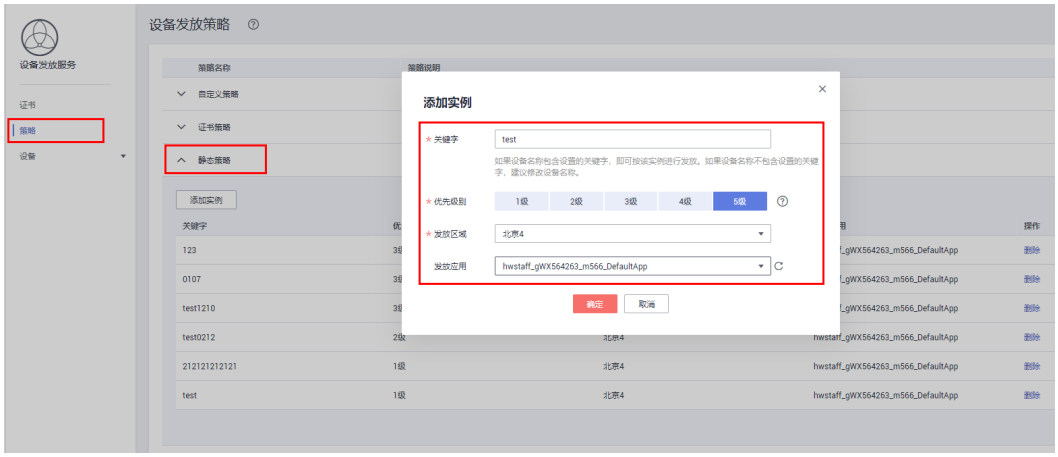
静态策略，即设备关键字模糊匹配的发放策略。每条静态策略实例指示匹配上该策略实例的设备将会被发放到该策略实例关联的设备接入实例的对应资源空间（即应用）下。

设备匹配静态策略实例的机制为：

1. 设备的发放策略为“静态策略”，设备的设备名称包含某一静态策略实例的关键字，即被认为该设备匹配上该条策略实例；
2. 一个设备最多匹配一条静态策略实例；
3. 当一个设备匹配上多条静态策略实例时，则以优先级最高的策略实例为优先。

操作步骤

步骤1 进入“策略”界面，单击展开“静态策略”，单击“添加实例”。



步骤2 按照下方参数说明填写关键参数信息后，单击“确定”。

参数名称	说明	示例
关键字	即设备名称中的关键字。设备发放时，如果设备名称包含设置的关键字，则可按该实例进行发放。	将设备名称携带Beijing的设备发放至华北-北京四的物联网平台。 • 设备名称：WaterMeter-Beijing0001、WaterMeter-Beijing0002 • 关键字：Beijing • 发放区域：华北-北京四 • 发放应用：beijing-app1
优先级别	发放策略的优先级，取值范围0 - 4，0为最低优先级。当一个设备符合多个发放策略时，按照优先级最高的策略实例发放。	
发放区域	发放到指定区域后，设备将接入对应区域的设备接入服务。 所选区域未开通设备接入服务时，如果确定添加实例，系统将自动为您开通设备接入服务。不同区域设备接入服务价格不同，收费详情请参考价格说明。	
发放应用	选择对应设备接入服务区域已创建的应用。在物联网平台中，设备由应用统一管理。 如果对应设备接入服务区域未创建应用，需要前往对应服务创建应用。	

----结束

11.4 设备

11.4.1 注册设备

注册设备用于将设备基本信息导入设备发放平台中，用于后续发放至不同的物联网平台，支持批量注册和单个注册。设备注册成功后，可在“设备-注册”中查看设备的详细信息。

说明

所需“产品”来自 设备接入服务 创建的产品，相关信息参见 **创建产品**。

单个/批量导入

步骤1 进入“设备-注册”界面，单击右上角“注册”或“批量注册”。

步骤2 按照下方参数说明填写关键参数信息后，完成注册。

参数名称	说明
产品	选择设备所属的产品名称。 批量导入时，需要填写产品ID。
设备标识码	设备唯一物理标识，如IMEI、MAC地址等，用于设备在接入物联网平台时携带该标识信息完成接入鉴权。
设备名称	设备发放时，设备名称将作为主要的筛选条件之一，建议按照统一的规则规划不同地区的设备名称。 例如：北京地区设备，设备名称都携带Beijing；香港地区设备，设备名称都携带Hongkong。
设备ID	设备ID默认为产品ID+“_”+设备标识码，用户也可自己定义。
发放策略	根据需求选择对应的发放策略类型，注册设备成功后，设备将按照对应的策略进行发放。

参数名称	说明
导入配置（LwM2M协议设备）	仅在设备使用LwM2M协议时需要填写以下参数： ● 引导服务端PSK：设备初次上电时，与设备发放服务建立DTLS连接，PSK验证通过后进行设备引导。 ● 连接服务端PSK：设备连接设备接入服务、与平台建立DTLS连接时用于验证。 **连接服务端PSK不需要烧录在设备中，设备首次上电后启动BootStrap流程时由平台下发。** ● 安全模式：支持安全模式DTLS/DTLS+和非安全模式。选择DTLS模式或DTLS+模式，设备将通过安全端口5684接入平台，否则将通过非安全端口5683接入。 ● 默认生命周期：设备开机后，如果没有归属物联网平台地址，且在默认生命周期内没有收到设备发放服务下发的地址，则根据注册的设备发放服务地址，向物联网平台发起请求。平台返回连接信息后，设备覆盖本地的注册的物联网平台地址，向指定的物联网平台地址发起注册。默认值为86400秒，最大值（7*86400秒）。
导入配置（MQTT协议设备）	仅在设备使用MQTT协议时需要填写以下参数： ● 安全模式：支持安全模式密钥模式和X.509证书模式。 ● 设备密钥：选择密钥模式时该参数选填，如果不填系统会返回密钥，或从设备详情获取。 ● 选择证书：选择X.509证书模式时该参数必填，选择当前注册设备所需要的证书。 ● 证书指纹：根据证书生成的唯一识别证书的标识。与“自注册开关”参数选填其一或都填。 ● 自注册开关：如果支持设备自注册，在设备首次认证时不会去认证设备ID和设备证书的关系。与“证书指纹”参数选填其一或都填。

----结束

MQTT 设备批量发放操作

本文以注册3个MQTT设备，发放到中国站“华北-北京四”为例演示如何发放MQTT设备。

- **前提条件**

  已在中国站“华北-北京四”的设备接入服务中创建应用。

- **操作步骤**

  a. 进入“策略”界面，单击展开“静态策略”，单击“添加实例”。

  b. 按照下方参数说明填写关键参数信息后，单击“确定”。

表 11-12 实例 1-发放到北京

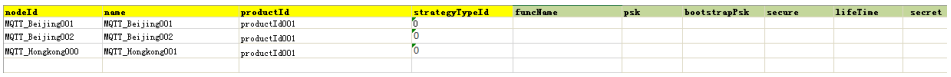
参数名称	说明
关键字	Beijing
优先级别	1级
发放区域	华北-北京四
发放应用	选择在“华北-北京四”的设备接入服务中创建的应用。

c. 进入“设备”界面，单击右上角“批量注册”。

d. 下载模版，按照下方参数说明分别填写三个设备的关键参数信息后，选择模板文件并单击“保存”。

参数名称	说明
nodeId	若有真实设备，填写为设备的IMEI或MAC地址；若没有真实设备，填写自定义字符串，不同设备的识别码不能重复。
name	分别填写为MQTT_Beijing001，MQTT_Beijing002，MQTT_Beijing003。
productId	填写所要注册的设备所属产品的产品ID。
strategyTypeId	策略类型，填写0（静态策略）。
其他参数	留空。

图 11-2 模板填写样例



nodeId	name	productId	strategyTypeId	funcName	psk	bootstrapPsk	secure	lifeTime	secret
MQTT_Beijing001	MQTT_Beijing001	productId001	0						
MQTT_Beijing002	MQTT_Beijing002	productId001	0						
MQTT_Hongkong000	MQTT_Hongkong001	productId001	0						

e. 在“操作记录”中查看注册结果，注册成功则继续下一步；注册失败则可单击“失败数”所在列的数字查看失败原因，修改模板后重新注册。

f. 设备初次上电，先接入到设备发放平台，随后通过Bootstrap流程引导设备获得目标物联网平台地址，完成设备发放。

LwM2M 设备批量发放操作

本文以注册3个LwM2M设备，发放到中国站“华北-北京四”为例演示如何发放LwM2M设备。

- **前提条件**
  - 已在中国站“华北-北京四”的设备接入服务中创建应用。
  - 获取终端节点。

区域名称	区域	终端节点（Endpoint）	端口	协议
华北-北京四	cn-north-4	iot-bs.cn-north-4.myhuaweicloud.com	5683	LwM2M
华北-北京四	cn-north-4	iot-bs.cn-north-4.myhuaweicloud.com	5684	LwM2M + DTLS

- **操作步骤**

  a. 进入“策略”界面，单击展开“静态策略”，单击“添加实例”。

  b. 按照下方参数说明填写关键参数信息后，单击“确定”。

  表 11-13 实例 1-发放到北京

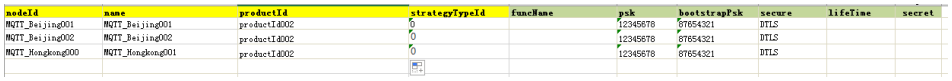
参数名称	说明
关键字	Beijing
优先级别	1级
发放区域	华北-北京四
发放应用	选择在“华北-北京四”的设备接入服务中创建的应用。

  c. 进入“设备”界面，单击右上角“批量注册”。

  d. 下载模版，按照下方参数说明分别填写三个设备的关键参数信息后，选择模板文件并单击“保存”。

参数名称	说明
nodeId	若有真实设备，填写为设备的IMEI或MAC地址；若没有真实设备，填写自定义字符串，不同设备的识别码不能重复。
name	分别填写为LwM2M_Beijing001，LwM2M_Beijing002，LwM2M_Beijing003。
productId	填写所要注册的设备所属产品的产品ID。
strategyTypeId	策略类型，填写0（静态策略）。
psk	可填写为12345678。
bootstrapPsk	可填写为87654321。
secure	填写为DTLS。
其他参数	留空。

图 11-3 模板填写样例



nodeId	name	productId	strategyTypeId	funcName	psk	bootstrapPsk	secure	lifeTime	secret
MQTT_Beijing001	MQTT_Beijing001	productId002			12345678	87654321	DTLS		
MQTT_Beijing002	MQTT_Beijing002	productId002			12345678	87654321	DTLS		
MQTT_Hongkong000	MQTT_Hongkong001	productId002			12345678	87654321	DTLS		

e. 在“操作记录”中查看注册结果，注册成功则继续下一步；注册失败则可单击“失败数”所在列的数字查看失败原因，修改模板后重新注册。

f. 设备初次上电，先接入到设备发放平台，随后通过Bootstrap流程引导设备获得目标物联网平台地址，完成设备发放。

11.4.2 注册组

MQTT证书接入的设备，可以在设备发放创建一个注册组，绑定对应的CA证书和自定义策略，可以实现批量设备的自注册，实现设备一键上电即可上云的动作，可在注册组详情中查看该注册组下所有的设备。

步骤1 进入“设备-注册组”界面，单击右上角“新增注册组”。

步骤2 按照下方参数说明填写关键参数信息后，完成创建。

参数名称	说明
注册组名称	注册组的唯一标识。
选择证书	用于和注册组绑定，同一个证书只能同时绑定一个注册组，不能同时绑定多个注册组。
发放策略	当前只支持“自定义策略”，同时需要选择所要运行的函数。

----结束

11.5 设备数据迁移

通过设备发放，将设备接入服务的设备业务数据实现跨实例迁移（比如由共享实例迁移到专享实例，或者从共享实例1迁移到共享实例2）。迁移时可以指定某些资源空间进行迁移，也可以指定迁移某产品下的一批设备。指定资源空间或者设备进行迁移会把必要数据（包括产品、CA证书、设备）都进行迁移。

⚠ 注意

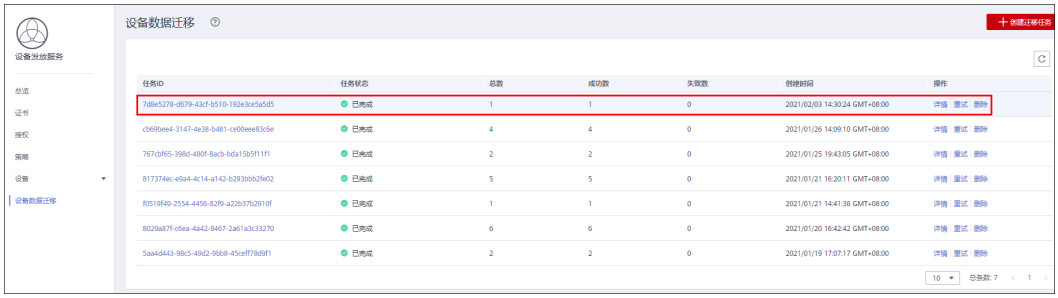
1. CA证书需要手动进行迁移。
2. 当迁移任务完成时，如果有设备迁移失败，可以重试迁移任务。当前运行中的迁移任务只能存在一个。
3. 按指定资源空间迁移时，最多支持10个资源空间同时迁移，且每个资源空间最大支持5万个设备迁移。
4. 按指定设备迁移时，最多支持1000个设备同时迁移。
5. 1个用户默认最多支持10条迁移任务记录。

按资源空间创建迁移任务

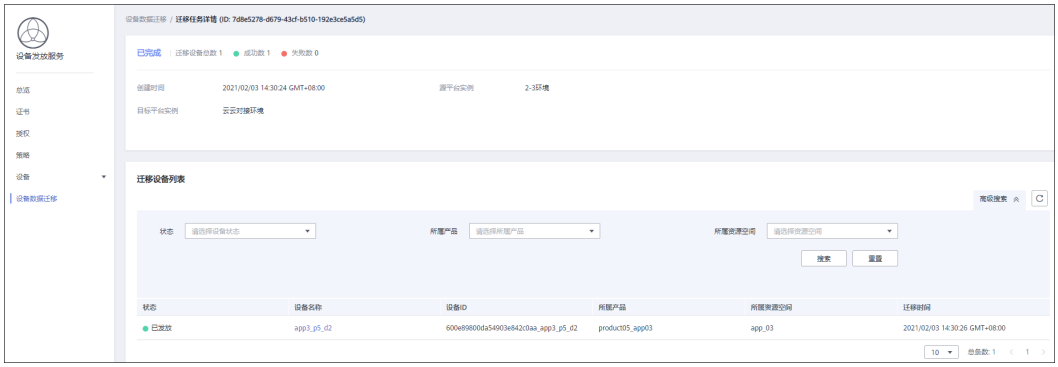
步骤1 进入“设备数据迁移”界面，单击右上角“创建迁移任务”。

步骤2 迁移方式选择“按资源空间整体迁移”，将所有必填选项选择完毕后，单击“确认创建”。

步骤3 在设备数据迁移任务界面，将会看到刚创建的任务，可以对任务进行查看详情、重试、停止或删除操作。



步骤4 单击“任务ID”或者“详情”，可以看到刚创建的任务的详细内容，并且可以使用“高级搜索”对迁移设备进行过滤查询操作。



步骤5 单击“设备名称”，可以跳转到设备迁移详情页，能够查看该设备的信息和设备的发放记录。

----结束

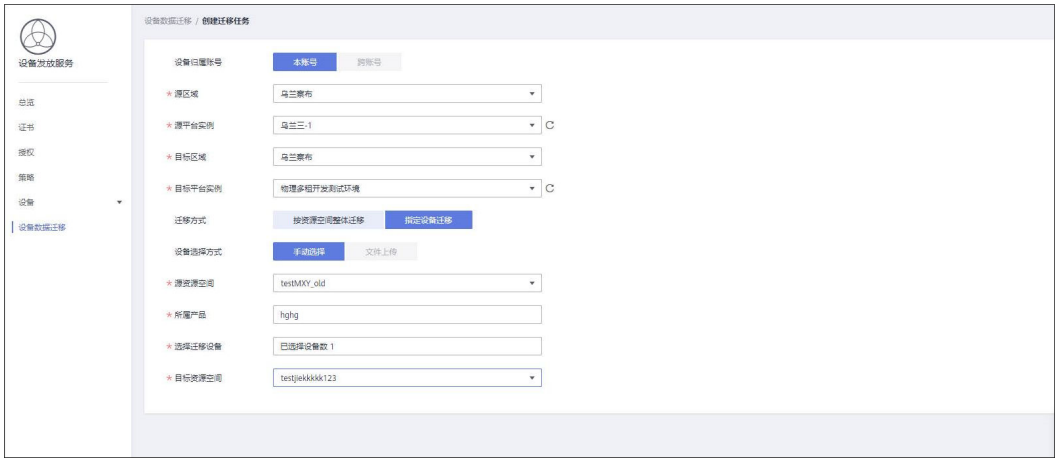
注意

1. 同一个资源空间不能创建多个迁移任务，如需对该资源空间下的数据进行重新迁移，要删除该资源空间关联的迁移任务。重新创建该资源空间的迁移任务后，相关数据迁移到新的目标实例，设备南向引导以最新迁移的目标平台实例为准下发地址。（删除迁移任务之后，如果没有创建新的迁移任务，设备南向无法引导。）
2. 当多个资源空间同时迁移时，若某资源空间下的产品、证书迁移失败只影响当前资源空间的迁移，其他资源空间正常执行。当产品和证书迁移失败时，可以选择通过重试迁移任务，尝试进行再次迁移。
3. 按资源空间迁移后，在源实例的设备接入控制台上查看不到迁移的资源空间。

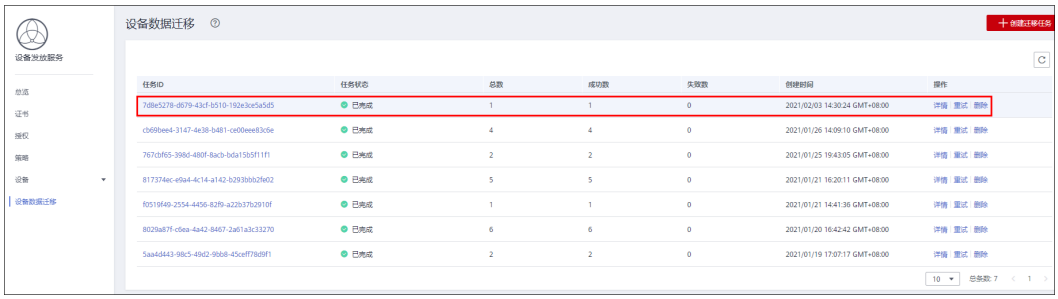
指定设备创建迁移任务

步骤1 进入“设备数据迁移”界面，单击右上角“创建迁移任务”。

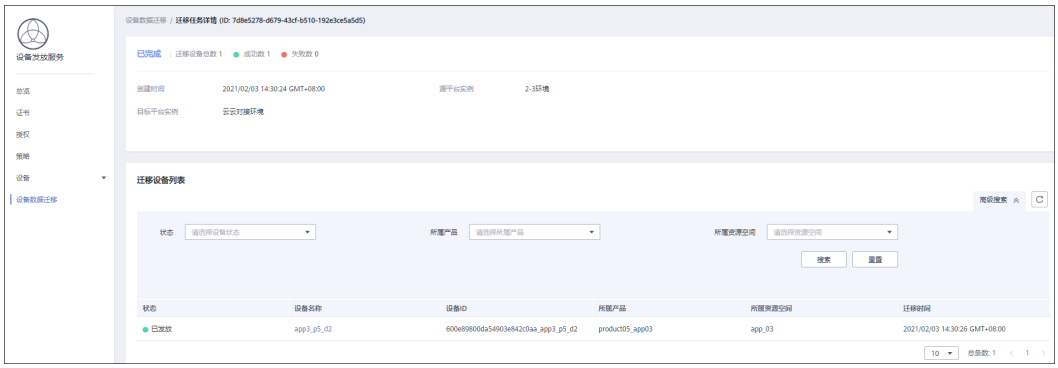
步骤2 迁移方式选择“指定设备迁移”，将所有必填选项选择完毕后，单击“确认创建”。



步骤3 在设备数据迁移任务界面，将会看到刚创建的任务，可以对任务进行查看详情、重试、停止和删除操作。



步骤4 单击“任务ID”或者“详情”，可以看到刚创建的任务的详细内容，并且可以使用“高级搜索”对迁移设备进行过滤查询操作。



说明

当迁移任务类型为“指定设备迁移”时，该类任务下所有设备的“所属产品”和“所属资源空间”均相同。因此在进行“高级搜索”时，搜索项“所属产品”和“所属资源空间”均为空。

步骤5 在设备迁移详情页，可以查看单个设备的信息和设备的发放记录。

----结束

注意

不同的设备可以根据不同的目标实例创建多个迁移任务，设备发放的发放数据以最新的一次的迁移任务为准。