08 API 参考

文档版本 01

发布日期 2024-07-02

华为技术有限公司

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。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 应用侧 API 参考

1.1 使用前必读

1.1.1 概述

物联网平台把自身丰富的管理能力通过API的形式对外开放，包括产品管理、设备管理、设备组管理、标签管理、设备CA证书管理、设备影子、设备命令、设备消息、设备属性、订阅管理、规则管理、批量任务等，帮助用户快速构筑基于物联网平台的行业应用。您可以根据本文档提供的API来使用物联网平台的服务，平台支持的全部API请参见**API列表**。

1.1.2 调用说明

物联网平台提供了RESTful（Representational State Transfer）风格API，支持您通过HTTPS请求调用，调用方法请参见**如何调用API**。

1.1.3 终端节点

终端节点（Endpoint）即调用API的请求地址，物联网平台的Endpoint请参见：**平台对接信息**中的接入地址。

1.1.4 约束与限制

- API的演进会保持前向兼容性，若接口升级了版本，其旧版本接口可以继续使用，但功能不再做增强，新增功能仅在新版本接口中提供。

- 应用在接收处理物联网平台发送的响应消息和推送消息时，需要兼容或忽略消息中的新增参数，不能因为消息中的新增参数导致应用的处理异常。
- 调用API的其他使用限制，请参见物联网平台的**使用限制**。

1.1.5 基本概念

- 账号

  用户注册华为云时的账号，账号对其所拥有的资源及云服务具有完全的访问权限，可以重置用户密码、分配用户权限等。由于账号是付费主体，为了确保账号安全，建议您不要直接使用账号进行日常管理工作，而是创建用户并使用他们进行日常管理工作。

- 用户

  由账号在IAM中创建的用户，是云服务的使用人员，具有身份凭证（密码和访问密钥）。

  在**我的凭证**下，您可以查看账号ID和用户ID。通常在调用API的鉴权过程中，您需要用到账号、用户和密码等信息。

- 区域（Region）

  从地理位置和网络时延维度划分，同一个Region内共享弹性计算、块存储、对象存储、VPC网络、弹性公网IP、镜像等公共服务。Region分为通用Region和专属Region，通用Region指面向公共租户提供通用云服务的Region；专属Region指只承载同一类业务或只面向特定租户提供业务服务的专用Region。

  详情请参见**区域和可用区**。

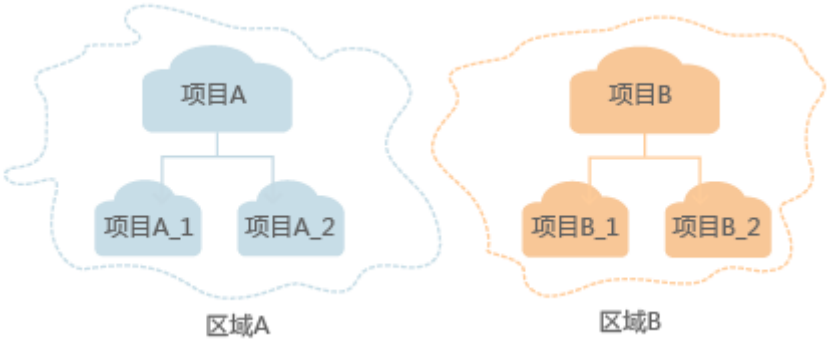
- 可用区（AZ，Availability Zone）

  一个AZ是一个或多个物理数据中心的集合，有独立的风火水电，AZ内逻辑上再将计算、网络、存储等资源划分成多个集群。一个Region中的多个AZ间通过高速光纤相连，以满足用户跨AZ构建高可用性系统的需求。

- 项目

  华为云的区域默认对应一个项目，这个项目由系统预置，用来隔离物理区域间的资源（计算资源、存储资源和网络资源），以默认项目为单位进行授权，用户可以访问您账号中该区域的所有资源。如果您希望进行更加精细的权限控制，可以在区域默认的项目中创建子项目，并在子项目中购买资源，然后以子项目为单位进行授权，使得用户仅能访问特定子项目中资源，使得资源的权限控制更加精确。

  图 1-1 项目隔离模型

- 企业项目

企业项目是项目的升级版，针对企业不同项目间资源的分组和管理，是逻辑隔离。企业项目中可以包含多个区域的资源，且项目中的资源可以迁入迁出。关于企业项目ID的获取及企业项目特性的详细信息，请参见《**企业管理服务用户指南**》。

- 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。详情请参见**资源空间**。

- 产品

  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。详情请参见**什么是产品模型**。

- 消息下发

  消息下发不依赖产品模型，提供给设备的单向通知，具有消息缓存功能；详情请参见**消息下发概述**。

- 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应用侧API接口向设备下发命令，以实现对设备的远程控制。详情请参见**命令下发概述**。

- 属性下发

  属性下发分为查询设备属性和修改属性参数两种，查询设备属性用于应用侧或平台主动获取设备属性数据，修改属性参数用于应用侧或平台设置设备属性值并同步到设备侧。详情请参见**属性下发概述**。

- AMQP队列管理

  AMQP（Advanced Message Queuing Protocol）即高级队列消息协议。用户可通过AMQP的客户端与IoT平台建立链接，来接收平台推送数据。详情请参见**AMQP订阅推送**。

- 数据流转

  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。详情请参见**订阅推送方式概述**。

- 设备影子

  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。详情请参见**设备影子**。

- 设备组（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。平台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。详情请参见**群组**。

- 标签

  对于拥有大量云资源的用户，可以通过给云资源打标签，快速查找具有某标签的云资源，可对这些资源标签统一进行检视、修改、删除等操作，方便用户对云资源的管理。详情请参见**标签概述**。

- 联动规则

  联动规则分为端侧规则和云端规则。端侧规则：在云端规则中，用户创建的规则的解析及执行均在云端完成，云平台需要判断条件是否满足并触发相应的设备联动操作。详情请参见**端侧规则**。云端规则：当用户设置云端规则时，物联网平台会判断是否满足规则触发条件，在条件满足时，平台会执行用户预设的动作。比如：事件告警、主题通知、设备命令下发等。详情请参见**云端规则**。

- 批量任务

  当需要对多个设备进行批量操作时，可以选择批量任务执行该操作。当前支持批量软固件升级、批量创建设备、批量修改设备、批量删除设备、批量冻结设备、批量解冻设备、批量创建命令、批量创建消息、批量配置设备影子和批量更新设备任务。

- OTA升级

  OTA升级指的是软固件升级，软件升级指升级设备的系统软件和应用软件，固件升级指升级设备硬件的底层“驱动程序”。升级方式均为将软/固件包上传到物联网平台或者关联用户OBS上的对象，设备从物联网平台或用户OBS获取软/固件包实现远程升级。详情请参见**OTA升级相关问题**。

1.2 如何调用 API

1.2.1 构造请求

本节介绍REST API请求的组成，并以调用物联网平台的**1.4.1.3 查询产品**和**1.4.1.1 创建产品**接口说明如何调用API，该API可用于获取用户的指定产品信息。

请求 URI

请求URI由如下部分组成。

{URI-scheme} :// {Endpoint} / {resource-path} ? {query-string}

尽管请求URI包含在请求消息头中，但大多数语言或框架都要求您从请求消息中单独传递它，所以在此单独强调。

- **URI-scheme**：表示用于传输请求的协议，当前所有API均采用**HTTPS**协议。
- **Endpoint**：指定承载REST服务端点的服务器域名或IP，不同服务不同区域的Endpoint不同，您可以从**平台对接信息**中的接入地址获取。
- **resource-path**：资源路径，也即API访问路径。从具体API的URI模块获取，例如“查询产品”API的resource-path为“/v5/iot/{project_id}/products/{product_id}”。
- **query-string**：查询参数，是可选部分，并不是每个API都有查询参数。查询参数前面需要带一个“？”，形式为“参数名=参数取值”，例如“limit=10”，表示查询不超过10条数据。

例如您需要在物联网平台获取在“华北-北京四”区域下的指定产品信息，则需使用“华北-北京四”区域的Endpoint（iotda.cn-north-4.myhuaweicloud.com），并在**1.4.1.3 查询产品**的URI部分找到resource-path（/v5/iot/{project_id}/products/{product_id}），拼接起来如下所示。

```
https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
```

> **说明**
>
> 为查看方便，在每个具体API的URI部分，只给出resource-path部分，并将请求方法写在一起。这是因为URI-scheme都是HTTPS，而Endpoint在同一个区域也相同，所以简洁起见将这两部分省略。

请求方法

HTTP请求方法（也称为操作或动词），它告诉服务你正在请求什么类型的操作。

- **GET**：请求服务器返回指定资源。
- **PUT**：请求服务器更新指定资源。
- **POST**：请求服务器新增资源或执行特殊操作。
- **DELETE**：请求服务器删除指定资源，如删除对象等。
- **HEAD**：请求服务器资源头部。
- **PATCH**：请求服务器更新资源的部分内容。当资源不存在的时候，PATCH可能会去创建一个新的资源。

在**1.4.1.3 查询产品**的URI部分，您可以看到其请求方法为“GET”，则其请求为：

```
GET https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
```

请求消息头

附加请求头字段，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。

如下公共消息头需要添加到请求中。

- **Content-Type**：消息体的类型（格式），必选，默认取值为“application/json;charset=utf-8”，有其他取值时会在具体接口中专门说明。
- **X-Auth-Token**：用户Token。当使用Token方式认证时，必须填充该字段，可通过调用**获取用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。

对于**1.4.1.3 查询产品**接口，由于需要认证，所以需要添加“Content-Type”和“X-Auth-Token”到头域，添加消息头后的请求如下所示。

```
GET https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
Content-Type: application/json
X-Auth-Token：******
```

请求消息体

请求消息体通常以结构化格式发出，与请求消息头中Content-type对应，传递除请求消息头之外的内容。若请求消息体中参数支持中文，则中文字符必须为UTF-8编码。

每个接口的请求消息体内容不同，也并不是每个接口都需要有请求消息体（或者说消息体为空），GET、DELETE操作类型的接口就不需要消息体，消息体具体内容需要根据具体接口而定。

对于**1.4.1.1 创建产品**接口，您可以从接口的请求部分看到所需的请求参数及参数说明。将消息体加入后的请求如下所示，加粗的斜体字段需要根据实际值填写，如：其中***name***为产品名称，***device_type***为设备类型，***protocol_type***为设备使用的协议类型。


```
POST https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/abab***cdcd/products
Content-Type: application/json
X-Auth-Token: ********

{
  "name" : "Thermometer",
  "device_type" : "Thermometer",
  "protocol_type" : "MQTT",
  "data_format" : "binary",
  "manufacturer_name" : "ABC",
  "industry" : "smartCity",
  "description" : "this is a thermometer produced by Huawei",
  "service_capabilities" : [ {
    "service_type" : "temperature",
    "service_id" : "temperature",
    "description" : "temperature",
    "properties" : [ {
      "unit" : "centigrade",
      "min" : "1",
      "method" : "R",
      "max" : "100",
      "data_type" : "decimal",
      "description" : "force",
      "step" : 0.1,
      "enum_list" : [ "string" ],
      "required" : true,
      "property_name" : "temperature",
      "max_length" : 100
    } ],
    "commands" : [ {
      "command_name" : "reboot",
      "responses" : [ {
        "response_name" : "ACK",
        "paras" : [ {
          "unit" : "km/h",
          "min" : "1",
          "max" : "100",
          "para_name" : "force",
          "data_type" : "string",
          "description" : "force",
          "step" : 0.1,
          "enum_list" : [ "string" ],
          "required" : false,
          "max_length" : 100
        } ]
      } ],
      "paras" : [ {
        "unit" : "km/h",
        "min" : "1",
        "max" : "100",
        "para_name" : "force",
        "data_type" : "string",
        "description" : "force",
        "step" : 0.1,
        "enum_list" : [ "string" ],
        "required" : false,
        "max_length" : 100
      } ]
    } ],
    "option" : "Mandatory"
  } ],
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
}
```

到此请求需要的内容已具备齐全，您可以使用**curl**、**Postman**或直接编写代码等方式发送请求调用API。

1.2.2 认证鉴权

调用接口有如下两种认证方式，您可以选择其中一种进行认证鉴权。

- Token认证：通过Token认证通用请求。
- AK/SK认证：通过AK（Access Key ID）/SK（Secret Access Key)加密调用请求。

Token 认证

说明

Token是服务端生成的一串字符串，作为客户端进行请求的一个令牌。第一次登录后，服务器生成一个Token并将此Token返回给客户端，以后客户端只需带上这个Token前来请求数据即可，无需再次带上用户名和密码。Token有效期是24小时，从客户端获取开始算起（24小时是相对时间），需要使用同一个Token鉴权时，建议缓存起来使用，避免频繁调用。在Token过期前，务必刷新Token或重新获取Token，否则Token过期后会在服务端鉴权失败。

如果您重新获取Token，不影响已有Token有效性。

Token在计算机系统中代表令牌（临时）的意思，拥有Token就代表拥有某种权限。Token认证就是在调用API的时候将Token加到请求消息头，从而通过身份认证，获得操作API的权限。

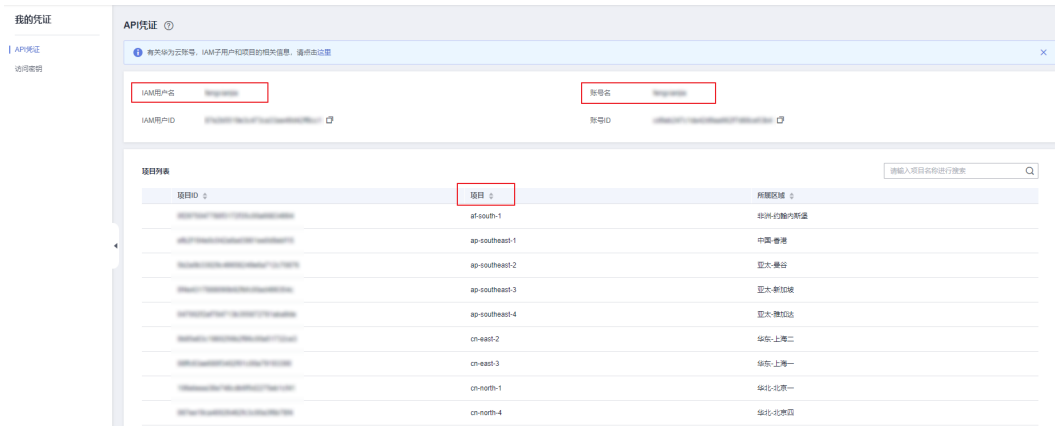
调用**获取IAM用户Token（使用密码）**接口获取Token，调用样例如下：

```
POST https://iam.cn-north-4.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",
                    "password": "********",
                    "domain": {
                        "name": "domainname"
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "projectname"
            }
        }
    }
}
```

注：“username”即IAM用户名、“password”即登录华为云密码、“domainname”即账号名，“projectname”即项目，您可以参考**我的凭证**页面获取。

图 1-2 API 凭证-获取凭证信息



接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。

获取Token后，再调用其他接口时，您需要在请求消息头中添加“X-Auth-Token”，其值为获取到的Token。例如Token值为“ABCDEFJ....”，则调用接口时将“X-Auth-Token: ABCDEFJ....”加到请求消息头即可，如下所示。

```
GET https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{project_id}/products/{product_id}
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

AK/SK 认证

说明

AK/SK签名认证方式仅支持消息体大小12M以内，12M以上的请求请使用Token认证。

AK/SK认证就是使用AK/SK对请求进行签名，在请求时将签名信息添加到消息头，从而通过身份认证。

- AK(Access Key ID)：访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。
- SK(Secret Access Key)：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。

使用AK/SK认证时，您可以基于签名算法使用AK/SK对请求进行签名，也可以使用专门的签名SDK对请求进行签名。详细的签名方法和SDK使用方法请参见： **AK/SK签名认证操作指导**。

说明

签名SDK只提供签名功能，与服务提供的SDK不同，使用时请注意。

1.2.3 返回结果

状态码

请求发送以后，您会收到响应，包含状态码、响应消息头和消息体。

状态码是一组从1xx到5xx的数字代码，状态码表示了请求响应的状态，完整的状态码列表请参见**1.7.1 状态码**。

对于1.4.1.1 创建产品接口，如果调用后返回状态码为“201”，则表示请求成功。

响应消息头

对应请求消息头，响应同样也有消息头，如“Content-type”等。

对于1.4.1.1 创建产品接口，平台会返回“Content-type”、Date等消息头。

响应消息体

响应消息体通常以结构化格式返回，与响应消息头中Content-type对应，传递除响应消息头之外的内容。

对于1.4.1.1 创建产品接口，返回如下消息体。为篇幅起见，这里只展示部分内容。

```
{
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "name" : "Thermometer",
  "device_type" : "Thermometer",
  "protocol_type" : "LWM2M",
  "data_format" : "binary",
  "manufacturer_name" : "ABC",
  "industry" : "smartCity",
  "description" : "this is a thermometer produced by Huawei",
    ......
}
```

当接口调用出错时，会返回错误码及错误信息说明，错误响应的Body体格式如下所示。

```
{
    "error_msg": "The format of message is error",
    "error_code": "IOTDA.013005"
}
```

其中，error_code表示错误码，error_msg表示错误描述信息。

1.3 API 概览

调用以下接口前，请先参考1.2.2 认证鉴权完成认证鉴权操作。

产品管理

API	说明
1.4.1.2 查询产品列表	查询已导入物联网平台的产品模型信息列表，了解产品模型的概要信息。
1.4.1.1 创建产品	创建产品。此接口仅创建了产品，没有创建和安装插件，如果需要对数据进行编解码，还需要在平台开发和安装插件。
1.4.1.3 查询产品	查询已导入物联网平台的指定产品模型详细信息，包括产品模型的服务、属性、命令等。

API	说明
1.4.1.4 修改产品	修改已导入物联网平台的指定产品模型，包括产品模型的服务、属性、命令等。此接口仅修改了产品，未修改和安装插件，如果修改了产品中的service定义，且在平台中有对应的插件，请修改并重新安装插件。
1.4.1.5 删除产品	删除已导入物联网平台的指定产品模型。

设备管理

API	说明
1.4.2.2 查询设备列表	查询物联网平台中的设备信息列表。
1.4.2.1 创建设备	在物联网平台注册一个设备，仅在注册后设备才可以接入物联网平台。
1.4.2.3 查询设备	查询物联网平台中指定设备的详细信息。
1.4.2.4 修改设备	修改物联网平台中指定设备的基本信息。
1.4.2.5 删除设备	在物联网平台上删除指定设备。若设备下连接了非直连设备，则必须把设备下的非直连设备都删除后，才能删除该设备。
1.4.2.6 重置设备密钥	重置设备密钥，携带指定密钥时平台将设备密钥重置为指定的密钥，不携带密钥时平台将自动生成一个新的随机密钥返回。
1.4.2.7 冻结设备	冻结设备，设备冻结后不能再连接上线，可以通过解冻设备接口解除设备冻结。注意，当前仅支持冻结与平台直连的设备。
1.4.2.8 解冻设备	解冻设备，解除冻结后，设备可以连接上线。
1.4.2.9 重置设备指纹	应用服务器可调用此接口重置设备指纹。携带指定设备指纹时将之重置为指定值；不携带时将之置空。
1.4.2.10 灵活搜索设备列表	应用服务器使用SQL语句调用该接口，灵活的搜索所需要的设备资源列表。
1.4.2.11 查询指定设备加入的设备组列表	应用服务器调用该接口查询设备加入的设备组列表。

设备消息

API	说明
1.4.3.2 查询设备消息	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。

API	说明
1.4.3.1 下发设备消息	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。
1.4.3.3 查询指定消息id的消息	查询指定消息id的消息。

设备命令

API	说明
1.4.4.1.1 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同步命令，以实现对设备的同步控制。
1.4.4.2.1 下发异步设备命令	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异步命令，以实现对设备的控制。
1.4.4.2.2 查询指定id的命令	可通过指定id查询某条特定命令。

设备属性

API	说明
1.4.5.2 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。
1.4.5.1 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发属性。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。

AMQP 队列管理

API	说明
1.4.6.2 查询AMQP列表	可调用此接口查询物联网平台中的AMQP队列信息列表。
1.4.6.1 创建AMQP队列	可调用此接口在物联网平台创建AMQP队列。您可以通过调用数据流转规则管理接口，将数据推送到此AMQP队列。
1.4.6.3 查询单个AMQP队列	可调用此接口查询物联网平台中指定队列的详细信息。

API	说明
1.4.6.4 删除AMQP队列	可调用此接口在物联网平台上删除指定AMQP队列。

接入凭证管理

API	说明
1.4.7.1 生成接入凭证	接入码是用于客户端使用AMQP等协议与平台建链的一个认证凭据，调用该接口将会生成一对新的接入码。

数据流转规则管理

API	说明
1.4.8.2 查询规则条件列表	可调用此接口查询物联网平台中设置的规则条件列表。
1.4.8.1 创建规则触发条件	可调用此接口在物联网平台创建一条规则触发条件。
1.4.8.3 查询规则条件	可调用此接口查询物联网平台中指定规则条件的配置信息。
1.4.8.4 修改规则触发条件	可调用此接口修改物联网平台中指定规则条件的配置参数。
1.4.8.5 删除规则触发条件	可调用此接口删除物联网平台中的指定规则条件。
1.4.8.7 查询规则动作列表	可调用此接口查询物联网平台中设置的规则动作列表。
1.4.8.6 创建规则动作	可调用此接口在物联网平台创建一条规则动作。
1.4.8.8 查询规则动作	可调用此接口查询物联网平台中指定规则动作的配置信息。
1.4.8.9 修改规则动作	可调用此接口修改物联网平台中指定规则动作。
1.4.8.10 删除规则动作	可调用此接口删除物联网平台中的指定规则动作。

流转数据

API	说明
1.4.9.1 设备状态变更通知	应用服务器在物联网平台创建设备状态变更通知规则后，当物联网平台中的设备状态发生变更时，平台会向应用服务器推送通知消息。

API	说明
1.4.9.2 设备属性上报通知	应用服务器在物联网平台创建了设备属性上报通知规则后，当设备上报属性数据时，平台会向应用服务器推送通知消息。
1.4.9.3 设备消息状态变更通知	应用服务器在物联网平台创建了消息状态变更通知规则后，当设备消息状态变更时，平台会向应用服务器推送通知消息。
1.4.9.4 批量任务状态变更通知	应用服务器在物联网平台创建了批量任务状态变更通知规则后，当批量任务状态变更时，平台会向应用服务器推送通知消息。
1.4.9.5 设备消息上报通知	应用服务器在物联网平台创建了设备消息上报通知规则后，当设备上报消息数据时，平台会向应用服务器推送通知消息。
1.4.9.6 设备添加通知	应用服务器在物联网平台创建了设备添加事件规则后，当物联网平台中的设备添加时，平台会向应用服务器推送通知消息。
1.4.9.7 设备更新通知	应用服务器在物联网平台创建了设备更新事件规则后，当物联网平台中的设备更新时，平台会向应用服务器推送通知消息。
1.4.9.8 设备删除通知	应用服务器在物联网平台创建了设备删除通知事件规则后，当物联网平台中的设备删除时，平台会向应用服务器推送通知消息。
1.4.9.9 产品添加通知	应用服务器在物联网平台创建了产品添加通知事件规则后，当物联网平台中的产品添加时，平台会向应用服务器推送通知消息。
1.4.9.10 产品更新通知	应用服务器在物联网平台创建了产品更新通知事件规则后，当物联网平台中的产品更新时，平台会向应用服务器推送通知消息。
1.4.9.11 产品删除通知	应用服务器在物联网平台创建了产品删除通知事件规则后，当物联网平台中的产品删除时，平台会向应用服务器推送通知消息。
1.4.9.12 设备异步命令状态变更通知	应用服务器在物联网平台创建了命令状态变更通知后，当命令状态变更时，平台会向应用服务器推送通知消息。

设备联动规则

API	说明
1.4.10.2 查询规则列表	查询物联网平台中设置的设备联动规则列表。
1.4.10.1 创建规则	在物联网平台创建一条设备联动规则。
1.4.10.4 查询规则	查询物联网平台中指定规则的配置信息。
1.4.10.3 修改规则	修改物联网平台中指定规则的配置。
1.4.10.5 删除规则	删除物联网平台中的指定规则。

API	说明
1.4.10.6 修改规则状态	修改物联网平台中指定规则的状态，激活或者去激活规则。

设备影子

API	说明
1.4.11.1 查询设备影子数据	查询指定设备的设备影子信息，包括对设备的配置信息（desired区）和设备最新上报的数据信息（reported区）。
1.4.11.2 配置设备影子预期数据	配置设备影子的预期数据（desired区），当设备上线时把数据下发给设备。设备影子的属性和产品模型耦合在一起，配置的预期属性需在产品模型中定义且method具有可写属性“W”才可下发。

设备组管理

API	说明
1.4.12.2 查询设备组列表	查询物联网平台中的设备组信息列表。
1.4.12.1 添加设备组	新建设备组，一个华为云账号下最多可有1,000个分组，包括父分组和子分组。
1.4.12.3 查询设备组	查询设备组详情。
1.4.12.4 修改设备组	修改指定设备组。
1.4.12.5 删除设备组	删除指定设备组。
1.4.12.6 管理设备组中的设备	管理设备组中的设备，包括添加设备到设备组和从设备组删除设备。单个设备组内最多添加20,000个设备。一个设备最多可以被添加到10个设备组中。
1.4.12.7 查询设备组设备列表	查询指定设备组下的设备列表。

标签管理

API	说明
1.4.13.1 绑定标签	为指定资源绑定标签。当前仅支持为设备绑定标签。
1.4.13.2 解绑标签	为指定资源解绑标签。当前仅支持为设备解绑标签。

API	说明
1.4.13.3 按标签查询资源	查询绑定了指定标签的资源。

资源空间管理

API	说明
1.4.14.1 查询资源空间列表	资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。应用服务器可以调用此接口查询资源空间列表。
1.4.14.2 创建资源空间	资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。应用服务器可以调用此接口创建资源空间。
1.4.14.3 查询资源空间	资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。应用服务器可以调用此接口查询指定资源空间详情。
1.4.14.4 删除资源空间	删除指定资源空间。删除资源空间属于高危操作，删除资源空间后，该空间下的产品、设备等资源将不可用，请谨慎操作！

批量任务

API	说明
1.4.15.2 查询批量任务列表	查询物联网平台中批量任务列表，每一个任务又包括具体的任务内容、任务状态、任务完成情况统计等。
1.4.15.1 创建批量任务	创建批量处理任务，对多个设备进行批量操作。当前支持批量软固件升级、批量创建设备、批量修改设备、批量删除设备、批量冻结设备、批量解冻设备、批量创建命令、批量创建消息、批量配置设备影子和批量更新设备任务。
1.4.15.3 查询批量任务	查询物联网平台中指定批量任务的信息，包括任务内容、任务状态、任务完成情况统计以及子任务列表等。
1.4.15.4 删除批量任务	应用服务器可调用此接口删除物联网平台中已经完成（状态为成功，失败，部分成功，已停止）的批量任务。
1.4.15.5 重试批量任务	应用服务器可调用此接口重试批量任务，目前只支持task_type为firmwareUpgrade，softwareUpgrade。如果task_id对应任务已经成功、停止、正在停止、等待中或初始化中，则不可以调用该接口。

API	说明
1.4.15.6 停止批量任务	应用服务器可调用此接口停止批量任务，目前只支持task_type为firmwareUpgrade，softwareUpgrade。如果task_id对应任务已经完成（成功、失败、部分成功，已经停止）或正在停止中，则不可以调用该接口。
1.4.15.7.2 查询批量任务文件列表	查询批量任务文件列表。
1.4.15.7.1 上传批量任务文件	上传批量任务文件，用于创建批量任务。当前支持批量创建设备任务、批量删除设备任务、批量冻结设备任务、批量解冻设备和批量更新设备任务的文件上传。
1.4.15.7.3 删除批量任务文件	删除批量任务文件。

设备 CA 证书管理

API	说明
1.4.16.2 获取设备CA证书列表	获取设备的CA证书列表。
1.4.16.1 上传设备CA证书	上传设备的CA证书。
1.4.16.3 删除设备CA证书	删除设备的CA证书。
1.4.16.5 验证设备CA证书	验证设备的CA证书，目的是为了验证用户持有设备CA证书的私钥。
1.4.16.4 更新CA证书	更新设备CA证书。

OTA 升级包管理

API	说明
1.4.17.1 创建OTA升级包	用户可调用此接口创建升级包关联OBS对象。
1.4.17.2 查询OTA升级包列表	用户可调用此接口查询关联OBS对象的升级包列表。
1.4.17.3 获取OTA升级包详情	用户可调用此接口查询关联OBS对象的升级包详情。
1.4.17.4 删除OTA升级包	用户可调用此接口删除关联OBS对象的升级包信息，不会删除OBS上对象。

广播消息

API	说明
1.4.18.1 下发广播消息	应用服务器可调用此接口向订阅了指定Topic的所有在线设备发布广播消息。

设备隧道管理

API	说明
1.4.19.1 创建设备隧道	应用服务器可调用此接口创建隧道。
1.4.19.2 查询设备所有隧道	应用服务器可调用此接口查询所有设备隧道。
1.4.19.3 查询设备隧道	应用服务器可调用此接口查询设备隧道详情。
1.4.19.4 关闭设备隧道	应用服务器可调用此接口关闭隧道。
1.4.19.5 删除设备隧道	应用服务器可调用此接口删除隧道。

设备代理

API	说明
1.4.22.1 创建设备代理	应用服务器可调用此接口设备代理。
1.4.22.2 查询设备代理列表	应用服务器可调用此接口查询所有设备代理。
1.4.22.3 查询设备代理详情	应用服务器可调用此接口查询设备代理详情。
1.4.22.4 修改设备代理	应用服务器可调用此接口修改设备代理。
1.4.22.5 删除设备代理	应用服务器可调用此接口删除设备代理。

网桥管理

API	说明
1.4.23.1 创建网桥	应用服务器可调用此接口网桥。
1.4.23.2 查询网桥列表	应用服务器可调用此接口查询所有网桥。
1.4.23.3 删除网桥	应用服务器可调用此接口删除网桥。
1.4.23.4 重置网桥密钥	应用服务器可调用此接口重置网桥密码。

设备策略管理

API	说明
1.4.24.1 创建设备策略	应用服务器可调用此接口创建设备策略。
1.4.24.2 查询设备策略列表	应用服务器可调用此接口查询所有设备策略。
1.4.24.3 删除设备策略	应用服务器可调用此接口删除设备策略。
1.4.24.4 查询设备策略详情	应用服务器可调用此接口查询设备策略详情。
1.4.24.5 更新设备策略信息	应用服务器可调用此接口更新设备策略信息。
1.4.24.6 绑定设备策略	应用服务器可调用此接口绑定设备策略。
1.4.24.7 解绑设备策略	应用服务器可调用此接口解绑设备策略。
1.4.24.8 查询设备策略绑定的目标列表	应用服务器可调用此接口查询该设备策略绑定的目标列表。

预调配模板管理

API	说明
1.4.25.1 创建预调配模板	应用服务器可调用此接口创建创建预调配模板。
1.4.25.2 查询预调配模板列表	应用服务器可调用此接口查询所有创建预调配模板。
1.4.25.3 删除预调配模板	应用服务器可调用此接口删除预调配模板。
1.4.25.4 查询预调配模板详情	应用服务器可调用此接口查询创建预调配模板详情。
1.4.25.5 更新指定id的预调配模板信息	应用服务器可调用此接口更新指定id的预调配模板信息。

自定义鉴权

API	说明
1.4.26.1 创建自定义鉴权	应用服务器可调用此接口创建自定义鉴权。
1.4.26.2 查询自定义鉴权列表	应用服务器可调用此接口查询所有自定义鉴权。

API	说明
1.4.26.3 删除自定义鉴权	应用服务器可调用此接口删除自定义鉴权。
1.4.26.4 查询自定义鉴权详情	应用服务器可调用此接口查询自定义鉴权详情。
1.4.26.5 更新指定id的自定义鉴权	应用服务器可调用此接口更新指定id的自定义鉴权。

1.4 API

1.4.1 产品管理

产品模型定义了该产品下所有设备具备的能力或特征，产品管理为应用服务器提供对物联网平台中产品模型的操作管理功能。

1.4.1.1 创建产品

功能介绍

应用服务器可调用此接口创建产品。此接口仅创建了产品，没有创建和安装插件，如果需要对数据进行编解码，还需要在平台开发和安装插件。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/products

表 1-1 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-2 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-3 请求 Body 参数

参数	是否必选	参数类型	描述
product_id	否	String	**参数说明**：产品ID，资源空间下唯一。用于资源空间下唯一标识一个产品。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。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
name	是	String	**参数说明**：产品名称。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
device_type	是	String	**参数说明**：设备类型。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
protocol_type	是	String	**参数说明**：设备使用的协议类型。 **取值范围**：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF，OPC-UA，OPC-DA，Other。

参数	是否必选	参数类型	描述
data_format	是	String	**参数说明**：设备上报数据的格式。 **取值范围**： • json：JSON格式 • binary：二进制码流格式 默认值json。 缺省值：**json**
service_capabilities	是	Array of **ServiceCapability** objects	**参数说明**：设备的服务能力列表。 **取值范围**：数组长度大小不超过500，内容大小不超过500k。
manufacturer_name	否	String	**参数说明**：厂商名称。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
industry	否	String	**参数说明**：设备所属行业。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
description	否	String	**参数说明**：产品的描述信息。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$！【】’‘“”（）？…~/等字符的组合。
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的产品归属到哪个资源空间下，否则创建的产品将会归属到**默认资源空间**下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

表 1-4 ServiceCapability

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID。注：产品内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。

参数	是否必选	参数类型	描述
service_type	是	String	**参数说明**：设备的服务类型。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
properties	否	Array of **ServiceProperty** objects	**参数说明**：设备服务支持的属性列表。 **取值范围**：数组长度大小不超过500。
commands	否	Array of **ServiceCommand** objects	**参数说明**：设备服务支持的命令列表。 **取值范围**：数组长度大小不超过500。
events	否	Array of **ServiceEvent** objects	**参数说明**：设备服务支持的事件列表。目前暂未支持自定义事件，创建/修改产品时无需定义该字段。 **取值范围**：数组长度大小不超过500。
description	否	String	**参数说明**：设备服务的描述信息。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$！【】’‘“”（）？…~/等字符的组合。
option	否	String	**参数说明**：指定设备服务是否必选。目前本字段为非功能性字段，仅起到标识作用。 **取值范围**： • Master：主服务 • Mandatory：必选服务 • Optional：可选服务 默认值为Optional。 缺省值：**Optional**

表 1-5 ServiceProperty

参数	是否必选	参数类型	描述
property_name	是	String	**参数说明**：设备属性名称。注：设备服务内不允许重复。属性名称作为设备影子JSON文档中的key不支持特殊字符：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。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	是	String	**参数说明**：设备属性的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	否	Boolean	**参数说明**：设备属性是否必选。默认为false。 缺省值：**false**
enum_list	否	Array of strings	**参数说明**：设备属性的枚举值列表。
min	否	String	**参数说明**：设备属性的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	否	String	**参数说明**：设备属性的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：设备属性的最大长度。
step	否	Double	**参数说明**：设备属性的步长。
unit	否	String	**参数说明**：设备属性的单位。 **取值范围**：长度不超过16。 最大长度：**16**

参数	是否必选	参数类型	描述
method	是	String	**参数说明**：设备属性的访问模式。 **取值范围**：RWE，RW，RE，WE，E，W，R。 • R：属性值可读 • W：属性值可写 • E：属性值可订阅，即属性值变化时上报事件
description	否	String	**参数说明**：设备属性的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$！【】’‘“”（）？…~/等字符的组合。
default_value	否	Object	**参数说明**：设备属性的默认值。如果设置了默认值，使用该产品创建设备时，会将该属性的默认值写入到该设备的设备影子预期数据中，待设备上线时将该属性默认值下发给设备。

表 1-6 ServiceCommand

参数	是否必选	参数类型	描述
command_name	是	String	**参数说明**：设备命令名称。注：设备服务内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令的参数列表。
responses	否	Array of **ServiceCommandResponse** objects	**参数说明**：设备命令的响应列表。

表 1-7 ServiceCommandResponse

参数	是否必选	参数类型	描述
response_name	是	String	**参数说明**：设备命令响应名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令响应的参数列表。

表 1-8 ServiceEvent

参数	是否必选	参数类型	描述
event_type	是	String	**参数说明**：设备事件类型。注：设备服务内不允许重复。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备事件的参数列表。

表 1-9 ServiceCommandPara

参数	是否必选	参数类型	描述
para_name	是	String	**参数说明**：参数的名称。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	是	String	**参数说明**：参数的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	否	Boolean	**参数说明**：参数是否必选。默认为false。 缺省值：**false**
enum_list	否	Array of strings	**参数说明**：参数的枚举值列表。

参数	是否必选	参数类型	描述
min	否	String	**参数说明**：参数的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	否	String	**参数说明**：参数的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：参数的最大长度。
step	否	Double	**参数说明**：参数的步长。
unit	否	String	**参数说明**：参数的单位。 **取值范围**：长度不超过16。 最大长度：**16**
description	否	String	**参数说明**：参数的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$!【】’‘“”（）？…~/等字符的组合。

响应参数

状态码： 201

表 1-10 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID。
app_name	String	资源空间名称。
product_id	String	产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。
name	String	产品名称。
device_type	String	设备类型。
protocol_type	String	设备使用的协议类型。取值范围：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF，OPC-UA，OPC-DA，Other。
data_format	String	设备上报数据的格式，取值范围：json，binary。

参数	参数类型	描述
manufacturer_name	String	厂商名称。
industry	String	设备所属行业。
description	String	产品的描述信息。
service_capabilities	Array of **ServiceCapability** objects	设备的服务能力列表。
create_time	String	在物联网平台创建产品的时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-11 ServiceCapability

参数	参数类型	描述
service_id	String	**参数说明**：设备的服务ID。注：产品内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
service_type	String	**参数说明**：设备的服务类型。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
properties	Array of **ServiceProperty** objects	**参数说明**：设备服务支持的属性列表。 **取值范围**：数组长度大小不超过500。
commands	Array of **ServiceCommand** objects	**参数说明**：设备服务支持的命令列表。 **取值范围**：数组长度大小不超过500。
events	Array of **ServiceEvent** objects	**参数说明**：设备服务支持的事件列表。目前暂未支持自定义事件，创建/修改产品时无需定义该字段。 **取值范围**：数组长度大小不超过500。
description	String	**参数说明**：设备服务的描述信息。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$!【】’ ‘“”（）？…~/等字符的组合。
option	String	**参数说明**：指定设备服务是否必选。目前本字段为非功能性字段，仅起到标识作用。 **取值范围**： • Master：主服务 • Mandatory：必选服务 • Optional：可选服务 默认值为Optional。 缺省值：**Optional**

表 1-12 ServiceProperty

参数	参数类型	描述
property_name	String	**参数说明**：设备属性名称。注：设备服务内不允许重复。属性名称作为设备影子JSON文档中的key不支持特殊字符：点(.)、dollar符号($)、空char(十六进制的ASCII码为00)，如果包含了以上特殊字符则无法正常刷新影子文档。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	String	**参数说明**：设备属性的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	Boolean	**参数说明**：设备属性是否必选。默认为false。 缺省值：**false**
enum_list	Array of strings	**参数说明**：设备属性的枚举值列表。
min	String	**参数说明**：设备属性的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	String	**参数说明**：设备属性的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	Integer	**参数说明**：设备属性的最大长度。
step	Double	**参数说明**：设备属性的步长。
unit	String	**参数说明**：设备属性的单位。 **取值范围**：长度不超过16。 最大长度：**16**
method	String	**参数说明**：设备属性的访问模式。 **取值范围**：RWE，RW，RE，WE，E，W，R。 ● R：属性值可读 ● W：属性值可写 ● E：属性值可订阅，即属性值变化时上报事件
description	String	**参数说明**：设备属性的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$!【】’‘“”（）？…~/等字符的组合。

参数	参数类型	描述
default_value	Object	**参数说明**：设备属性的默认值。如果设置了默认值，使用该产品创建设备时，会将该属性的默认值写入到该设备的设备影子预期数据中，待设备上线时将该属性默认值下发给设备。

表 1-13 ServiceCommand

参数	参数类型	描述
command_name	String	**参数说明**：设备命令名称。注：设备服务内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备命令的参数列表。
responses	Array of **ServiceCommandResponse** objects	**参数说明**：设备命令的响应列表。

表 1-14 ServiceCommandResponse

参数	参数类型	描述
response_name	String	**参数说明**：设备命令响应名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备命令响应的参数列表。

表 1-15 ServiceEvent

参数	参数类型	描述
event_type	String	**参数说明**：设备事件类型。注：设备服务内不允许重复。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

参数	参数类型	描述
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备事件的参数列表。

表 1-16 ServiceCommandPara

参数	参数类型	描述
para_name	String	**参数说明**：参数的名称。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	String	**参数说明**：参数的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	Boolean	**参数说明**：参数是否必选。默认为false。 缺省值：**false**
enum_list	Array of strings	**参数说明**：参数的枚举值列表。
min	String	**参数说明**：参数的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	String	**参数说明**：参数的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	Integer	**参数说明**：参数的最大长度。
step	Double	**参数说明**：参数的步长。
unit	String	**参数说明**：参数的单位。 **取值范围**：长度不超过16。 最大长度：**16**
description	String	**参数说明**：参数的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$! 【】’ ‘“”（）？…~/等字符的组合。

请求示例

创建一个产品，名称为Thermometer，服务为temperature，包含属性temperature和命令reboot。

```
POST https://{endpoint}/v5/iot/{project_id}/products

{
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "name" : "Thermometer，",
  "device_type" : "Thermometer，",
  "protocol_type" : "MQTT",
  "data_format" : "json",
  "service_capabilities" : [ {
    "service_id" : "temperature，",
    "service_type" : "temperature",
    "properties" : [ {
      "property_name" : "temperature",
      "data_type" : "decimal",
      "required" : true,
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "centigrade",
      "method" : "RW",
      "description" : "force",
      "default_value" : {
        "color" : "red",
        "size" : 1
      }
    } ],
    "commands" : [ {
      "command_name" : "reboot",
      "paras" : [ {
        "para_name" : "force",
        "data_type" : "string",
        "required" : false,
        "enum_list" : null,
        "min" : "1",
        "max" : "100",
        "max_length" : 100,
        "step" : 0.1,
        "unit" : "km/h",
        "description" : "force"
      } ],
      "responses" : [ {
        "response_name" : "ACK",
        "paras" : [ {
          "para_name" : "force",
          "data_type" : "string",
          "required" : false,
          "enum_list" : null,
          "min" : "1",
          "max" : "100",
          "max_length" : 100,
          "step" : 0.1,
          "unit" : "km/h",
          "description" : "force"
        } ]
      } ]
    } ],
    "description" : "temperature",
    "option" : "Mandatory"
  } ],
  "manufacturer_name" : "ABC",
  "industry" : "smartCity",
  "description" : "this is a thermometer produced by Huawei",
```



```
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
}
```

响应示例

状态码： 201

Created

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "app_name" : "testAPP01",
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "name" : "Thermometer",
  "device_type" : "Thermometer",
  "protocol_type" : "MQTT",
  "data_format" : "json",
  "manufacturer_name" : "ABC",
  "industry" : "smartCity",
  "description" : "this is a thermometer produced by Huawei",
  "service_capabilities" : [ {
    "service_id" : "temperature",
    "service_type" : "temperature",
    "properties" : [ {
      "property_name" : "temperature",
      "required" : true,
      "data_type" : "decimal",
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "centigrade",
      "method" : "RW",
      "description" : "force",
      "default_value" : {
        "color" : "red",
        "size" : 1
      }
    } ],
    "commands" : [ {
      "command_name" : "reboot",
      "paras" : [ {
        "para_name" : "force",
        "required" : false,
        "data_type" : "string",
        "enum_list" : null,
        "min" : "1",
        "max" : "100",
        "max_length" : 100,
        "step" : 0.1,
        "unit" : "km/h",
        "description" : "force"
      } ],
      "responses" : [ {
        "response_name" : "ACK",
        "paras" : [ {
          "para_name" : "force",
          "required" : false,
          "data_type" : "string",
          "enum_list" : null,
          "min" : "1",
          "max" : "100",
          "max_length" : 100,
          "step" : 0.1,
          "unit" : "km/h",
          "description" : "force"
        } ]
      } ]
```



```
  } ],
  "description" : "temperature",
  "option" : "Mandatory"
 } ],
 "create_time" : "20190303T081011Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建一个产品，名称为Thermometer，服务为temperature，包含属性temperature和命令reboot。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateProductSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateProductRequest request = new CreateProductRequest();
        AddProduct body = new AddProduct();
        List<ServiceCommandPara> listResponsesParas = new ArrayList<>();
        listResponsesParas.add(
            new ServiceCommandPara()
                .withParaName("force")
                .withDataType("string")
                .withRequired(false)
                .withMin("1")
                .withMax("100")
```

```
        .withMaxLength(100)
        .withStep((double)0.1)
        .withUnit("km/h")
        .withDescription("force")
);
List<ServiceCommandResponse> listCommandsResponses = new ArrayList<>();
listCommandsResponses.add(
    new ServiceCommandResponse()
        .withResponseName("ACK")
        .withParas(listResponsesParas)
);
List<ServiceCommandPara> listCommandsParas = new ArrayList<>();
listCommandsParas.add(
    new ServiceCommandPara()
        .withParaName("force")
        .withDataType("string")
        .withRequired(false)
        .withMin("1")
        .withMax("100")
        .withMaxLength(100)
        .withStep((double)0.1)
        .withUnit("km/h")
        .withDescription("force")
);
List<ServiceCommand> listServiceCapabilitiesCommands = new ArrayList<>();
listServiceCapabilitiesCommands.add(
    new ServiceCommand()
        .withCommandName("reboot")
        .withParas(listCommandsParas)
        .withResponses(listCommandsResponses)
);
List<ServiceProperty> listServiceCapabilitiesProperties = new ArrayList<>();
listServiceCapabilitiesProperties.add(
    new ServiceProperty()
        .withPropertyName("temperature")
        .withDataType("decimal")
        .withRequired(true)
        .withMin("1")
        .withMax("100")
        .withMaxLength(100)
        .withStep((double)0.1)
        .withUnit("centigrade")
        .withMethod("RW")
        .withDescription("force")
        .withDefaultValue("{\"color\":\"red\",\"size\":1}")
);
List<ServiceCapability> listbodyServiceCapabilities = new ArrayList<>();
listbodyServiceCapabilities.add(
    new ServiceCapability()
        .withServiceId("temperature,")
        .withServiceType("temperature")
        .withProperties(listServiceCapabilitiesProperties)
        .withCommands(listServiceCapabilitiesCommands)
        .withDescription("temperature")
        .withOption("Mandatory")
);
body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
body.withDescription("this is a thermometer produced by Huawei");
body.withIndustry("smartCity");
body.withManufacturerName("ABC");
body.withServiceCapabilities(listbodyServiceCapabilities);
body.withDataFormat("json");
body.withProtocolType("MQTT");
body.withDeviceType("Thermometer,");
body.withName("Thermometer,");
body.withProductId("5ba24f5ebbe8f56f5a14f605");
request.withBody(body);
try {
    CreateProductResponse response = client.createProduct(request);
```



```
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建一个产品，名称为Thermometer，服务为temperature，包含属性temperature和命令reboot。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateProductRequest()
        listParasResponses = [
            ServiceCommandPara(
                para_name="force",
                data_type="string",
                required=False,
                min="1",
                max="100",
                max_length=100,
                step=0.1,
                unit="km/h",
                description="force"
            )
        ]
        listResponsesCommands = [
            ServiceCommandResponse(
                response_name="ACK",
                paras=listParasResponses
```



```
        )
    ]
    listParasCommands = [
        ServiceCommandPara(
            para_name="force",
            data_type="string",
            required=False,
            min="1",
            max="100",
            max_length=100,
            step=0.1,
            unit="km/h",
            description="force"
        )
    ]
    listCommandsServiceCapabilities = [
        ServiceCommand(
            command_name="reboot",
            paras=listParasCommands,
            responses=listResponsesCommands
        )
    ]
    listPropertiesServiceCapabilities = [
        ServiceProperty(
            property_name="temperature",
            data_type="decimal",
            required=True,
            min="1",
            max="100",
            max_length=100,
            step=0.1,
            unit="centigrade",
            method="RW",
            description="force",
            default_value="{\"color\":\"red\",\"size\":1}"
        )
    ]
    listServiceCapabilitiesbody = [
        ServiceCapability(
            service_id="temperature,",
            service_type="temperature",
            properties=listPropertiesServiceCapabilities,
            commands=listCommandsServiceCapabilities,
            description="temperature",
            option="Mandatory"
        )
    ]
    request.body = AddProduct(
        app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
        description="this is a thermometer produced by Huawei",
        industry="smartCity",
        manufacturer_name="ABC",
        service_capabilities=listServiceCapabilitiesbody,
        data_format="json",
        protocol_type="MQTT",
        device_type="Thermometer,",
        name="Thermometer,",
        product_id="5ba24f5ebbe8f56f5a14f605"
    )
    response = client.create_product(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

Go

创建一个产品，名称为Thermometer，服务为temperature，包含属性temperature和命令reboot。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateProductRequest{}
    requiredParas:= false
    minParas:= "1"
    maxParas:= "100"
    maxLengthParas:= int32(100)
    stepParas:= float64(0.1)
    unitParas:= "km/h"
    descriptionParas:= "force"
    var listParasResponses = []model.ServiceCommandPara{
        {
            ParaName: "force",
            DataType: "string",
            Required: &requiredParas,
            Min: &minParas,
            Max: &maxParas,
            MaxLength: &maxLengthParas,
            Step: &stepParas,
            Unit: &unitParas,
            Description: &descriptionParas,
        },
    }
    var listResponsesCommands = []model.ServiceCommandResponse{
        {
            ResponseName: "ACK",
            Paras: &listParasResponses,
        },
    }
```



```
requiredParas1:= false
minParas1:= "1"
maxParas1:= "100"
maxLengthParas1:= int32(100)
stepParas1:= float64(0.1)
unitParas1:= "km/h"
descriptionParas1:= "force"
var listParasCommands = []model.ServiceCommandPara{
    {
        ParaName: "force",
        DataType: "string",
        Required: &requiredParas1,
        Min: &minParas1,
        Max: &maxParas1,
        MaxLength: &maxLengthParas1,
        Step: &stepParas1,
        Unit: &unitParas1,
        Description: &descriptionParas1,
    },
}
var listCommandsServiceCapabilities = []model.ServiceCommand{
    {
        CommandName: "reboot",
        Paras: &listParasCommands,
        Responses: &listResponsesCommands,
    },
}
requiredProperties:= true
minProperties:= "1"
maxProperties:= "100"
maxLengthProperties:= int32(100)
stepProperties:= float64(0.1)
unitProperties:= "centigrade"
descriptionProperties:= "force"
defaultValueProperties:= "{\"color\":\"red\",\"size\":1}"
var defaultValuePropertiesInterface interface{} = defaultValueProperties
var listPropertiesServiceCapabilities = []model.ServiceProperty{
    {
        PropertyName: "temperature",
        DataType: "decimal",
        Required: &requiredProperties,
        Min: &minProperties,
        Max: &maxProperties,
        MaxLength: &maxLengthProperties,
        Step: &stepProperties,
        Unit: &unitProperties,
        Method: "RW",
        Description: &descriptionProperties,
        DefaultValue: &defaultValuePropertiesInterface,
    },
}
descriptionServiceCapabilities:= "temperature"
optionServiceCapabilities:= "Mandatory"
var listServiceCapabilitiesbody = []model.ServiceCapability{
    {
        ServiceId: "temperature,",
        ServiceType: "temperature",
        Properties: &listPropertiesServiceCapabilities,
        Commands: &listCommandsServiceCapabilities,
        Description: &descriptionServiceCapabilities,
        Option: &optionServiceCapabilities,
    },
}
appIdAddProduct:= "jeQDJQZltU8iKgFFoW060F5SGZka"
descriptionAddProduct:= "this is a thermometer produced by Huawei"
industryAddProduct:= "smartCity"
manufacturerNameAddProduct:= "ABC"
productIdAddProduct:= "5ba24f5ebbe8f56f5a14f605"
request.Body = &model.AddProduct{
```



```
        AppId: &appIdAddProduct,
        Description: &descriptionAddProduct,
        Industry: &industryAddProduct,
        ManufacturerName: &manufacturerNameAddProduct,
        ServiceCapabilities: listServiceCapabilitiesbody,
        DataFormat: "json",
        ProtocolType: "MQTT",
        DeviceType: "Thermometer,",
        Name: "Thermometer,",
        ProductId: &productIdAddProduct,
    }
    response, err := client.CreateProduct(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.1.2 查询产品列表

功能介绍

应用服务器可调用此接口查询已导入物联网平台的产品模型信息列表，了解产品模型的概要信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/products

表 1-17 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-18 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的产品列表，不携带该参数则会查询该用户下所有产品列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
product_name	否	String	**参数说明**：产品名称。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-19 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-20 响应 Body 参数

参数	参数类型	描述
products	Array of **ProductSummary** objects	产品信息列表。
page	**Page** object	查询结果的分页信息。

表 1-21 ProductSummary

参数	参数类型	描述
app_id	String	资源空间ID。
app_name	String	资源空间名称。
product_id	String	产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。
name	String	产品名称。
device_type	String	设备类型。
protocol_type	String	设备使用的协议类型。取值范围：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF，OPC-UA，OPC-DA，Other。
data_format	String	设备上报数据的格式，取值范围：json，binary。
manufacturer_name	String	厂商名称。
industry	String	设备所属行业。
description	String	产品的描述信息。
create_time	String	在物联网平台创建产品的时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-22 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

列表查询所有产品。

```
GET https://{endpoint}/v5/iot/{project_id}/products
```

响应示例

状态码： 200

Successful response

```
{
  "products" : [ {
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "app_name" : "testAPP01",
    "product_id" : "5ba24f5ebbe8f56f5a14f605",
    "name" : "Thermometer",
    "device_type" : "Thermometer",
    "protocol_type" : "MQTT",
    "data_format" : "json",
    "manufacturer_name" : "ABC",
    "industry" : "smartCity",
    "description" : "this is a thermometer produced by Huawei",
    "create_time" : "20190303T081011Z"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListProductsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
```

```
            .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
            .withAk(ak)
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        ListProductsRequest request = new ListProductsRequest();
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withAppId("<app_id>");
        request.withProductName("<product_name>");
        request.withOffset(<offset>);
        try {
            ListProductsResponse response = client.listProducts(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListProductsRequest()
        request.limit = <limit>
```



```
        request.marker = "<marker>"
        request.app_id = "<app_id>"
        request.product_name = "<product_name>"
        request.offset = <offset>
        response = client.list_products(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListProductsRequest{}
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    productNameRequest:= "<product_name>"
    request.ProductName = &productNameRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListProducts(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Successful response
400	Bad Request
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.1.3 查询产品

功能介绍

应用服务器可调用此接口查询已在物联网平台的创建的指定产品模型详细信息，包括产品模型的服务、属性、命令等。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/products/{product_id}

表 1-23 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
product_id	是	String	**参数说明**：产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

表 1-24 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数，指定要查询的产品属于哪个资源空间；若不携带，则优先取**默认资源空间**下产品，如默认资源空间下无对应产品，则按照产品创建时间取最早创建产品。如果用户存在多资源空间，同时又不想携带该参数，可以联系华为技术支持对用户数据做资源空间合并。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-25 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-26 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID。
app_name	String	资源空间名称。
product_id	String	产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。
name	String	产品名称。
device_type	String	设备类型。
protocol_type	String	设备使用的协议类型。取值范围：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF，OPC-UA，OPC-DA，Other。
data_format	String	设备上报数据的格式，取值范围：json，binary。
manufacturer_name	String	厂商名称。
industry	String	设备所属行业。
description	String	产品的描述信息。
service_capabilities	Array of **ServiceCapability** objects	设备的服务能力列表。
create_time	String	在物联网平台创建产品的时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-27 ServiceCapability

参数	参数类型	描述
service_id	String	**参数说明**：设备的服务ID。注：产品内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
service_type	String	**参数说明**：设备的服务类型。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
properties	Array of **ServiceProperty** objects	**参数说明**：设备服务支持的属性列表。 **取值范围**：数组长度大小不超过500。
commands	Array of **ServiceCommand** objects	**参数说明**：设备服务支持的命令列表。 **取值范围**：数组长度大小不超过500。

参数	参数类型	描述
events	Array of ServiceEvent objects	**参数说明**：设备服务支持的事件列表。目前暂未支持自定义事件，创建/修改产品时无需定义该字段。 **取值范围**：数组长度大小不超过500。
description	String	**参数说明**：设备服务的描述信息。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$!【】’ ‘“”（）？…~/等字符的组合。
option	String	**参数说明**：指定设备服务是否必选。目前本字段为非功能性字段，仅起到标识作用。 **取值范围**： ● Master：主服务 ● Mandatory：必选服务 ● Optional：可选服务 默认值为Optional。 缺省值：**Optional**

表 1-28 ServiceProperty

参数	参数类型	描述
property_name	String	**参数说明**：设备属性名称。注：设备服务内不允许重复。属性名称作为设备影子JSON文档中的key不支持特殊字符：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。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	String	**参数说明**：设备属性的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	Boolean	**参数说明**：设备属性是否必选。默认为false。 缺省值：**false**
enum_list	Array of strings	**参数说明**：设备属性的枚举值列表。
min	String	**参数说明**：设备属性的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	String	**参数说明**：设备属性的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**

参数	参数类型	描述
max_length	Integer	**参数说明**：设备属性的最大长度。
step	Double	**参数说明**：设备属性的步长。
unit	String	**参数说明**：设备属性的单位。 **取值范围**：长度不超过16。 最大长度：**16**
method	String	**参数说明**：设备属性的访问模式。 **取值范围**：RWE，RW，RE，WE，E，W，R。 ● R：属性值可读 ● W：属性值可写 ● E：属性值可订阅，即属性值变化时上报事件
description	String	**参数说明**：设备属性的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$! 【 】’ ‘ “ ” （ ）？ …~/等字符的组合。
default_value	Object	**参数说明**：设备属性的默认值。如果设置了默认值，使用该产品创建设备时，会将该属性的默认值写入到该设备的设备影子预期数据中，待设备上线时将该属性默认值下发给设备。

表 1-29 ServiceCommand

参数	参数类型	描述
command_name	String	**参数说明**：设备命令名称。注：设备服务内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备命令的参数列表。
responses	Array of **ServiceCommandResponse** objects	**参数说明**：设备命令的响应列表。

表 1-30 ServiceCommandResponse

参数	参数类型	描述
response_name	String	**参数说明**：设备命令响应名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备命令响应的参数列表。

表 1-31 ServiceEvent

参数	参数类型	描述
event_type	String	**参数说明**：设备事件类型。注：设备服务内不允许重复。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备事件的参数列表。

表 1-32 ServiceCommandPara

参数	参数类型	描述
para_name	String	**参数说明**：参数的名称。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	String	**参数说明**：参数的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	Boolean	**参数说明**：参数是否必选。默认为false。 缺省值：**false**
enum_list	Array of strings	**参数说明**：参数的枚举值列表。
min	String	**参数说明**：参数的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**

参数	参数类型	描述
max	String	**参数说明**：参数的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	Integer	**参数说明**：参数的最大长度。
step	Double	**参数说明**：参数的步长。
unit	String	**参数说明**：参数的单位。 **取值范围**：长度不超过16。 最大长度：**16**
description	String	**参数说明**：参数的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$！【】’ ‘“”（）？…~/等字符的组合。

请求示例

查询指定产品详情。

```
GET https://{endpoint}/v5/iot/{project_id}/products/{product_id}
```

响应示例

状态码： 200

Successful response

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "app_name" : "testAPP01",
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "name" : "Thermometer",
  "device_type" : "Thermometer",
  "protocol_type" : "MQTT",
  "data_format" : "json",
  "manufacturer_name" : "ABC",
  "industry" : "smartCity",
  "description" : "this is a thermometer produced by Huawei",
  "service_capabilities" : [ {
    "service_id" : "temperature",
    "service_type" : "temperature",
    "properties" : [ {
      "property_name" : "temperature",
      "required" : true,
      "data_type" : "decimal",
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "centigrade",
      "method" : "RW",
      "description" : "force",
      "default_value" : {
```



```
      "color" : "red",
      "size" : 1
    }
   } ],
   "commands" : [ {
    "command_name" : "reboot",
    "paras" : [ {
     "para_name" : "force",
     "required" : false,
     "data_type" : "string",
     "enum_list" : null,
     "min" : "1",
     "max" : "100",
     "max_length" : 100,
     "step" : 0.1,
     "unit" : "km/h",
     "description" : "force"
    } ],
    "responses" : [ {
     "response_name" : "ACK",
     "paras" : [ {
      "para_name" : "force",
      "required" : false,
      "data_type" : "string",
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "km/h",
      "description" : "force"
     } ]
    } ]
   } ],
   "events" : [ {
    "event_type" : "reboot",
    "paras" : [ {
     "para_name" : "force",
     "required" : false,
     "data_type" : "string",
     "enum_list" : null,
     "min" : "1",
     "max" : "100",
     "max_length" : 100,
     "step" : 0.1,
     "unit" : "km/h",
     "description" : "force"
    } ]
   } ],
   "description" : "temperature",
   "option" : "Mandatory"
  } ],
  "create_time" : "20190303T081011Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
```

```
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowProductSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowProductRequest request = new ShowProductRequest();
        request.withAppId("<app_id>");
        try {
            ShowProductResponse response = client.showProduct(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
```

```
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowProductRequest()
        request.app_id = "<app_id>"
        response = client.show_product(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowProductRequest{}
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    response, err := client.ShowProduct(request)
    if err == nil {
```

```
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Successful response
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.1.4 修改产品

功能介绍

应用服务器可调用此接口修改已导入物联网平台的指定产品模型，包括产品模型的服务、属性、命令等。此接口仅修改了产品，未修改和安装插件，如果修改了产品中的service定义，且在平台中有对应的插件，请修改并重新安装插件。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/products/{product_id}

表 1-33 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

参数	是否必选	参数类型	描述
product_id	是	String	**参数说明**：产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-34 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-35 请求 Body 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数，指定要修改的产品属于哪个资源空间；若不携带，则优先修改**默认资源空间**下产品，如默认资源空间下无对应产品，则按照产品创建时间修改最早创建产品。如果用户存在多资源空间，同时又不想携带该参数，可以联系华为技术支持对用户数据做资源空间合并。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
name	否	String	**参数说明**：产品名称。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
device_type	否	String	**参数说明**：设备类型。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
protocol_type	否	String	**参数说明**：设备使用的协议类型。注：禁止其他协议类型修改为CoAP。 **取值范围**：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF，OPC-UA，OPC-DA，Other。
data_format	否	String	**参数说明**：设备上报数据的格式。 **取值范围**： • json：JSON格式 • binary：二进制码流格式
service_capabilities	否	Array of **ServiceCapability** objects	**参数说明**：设备的服务能力列表。
manufacturer_name	否	String	**参数说明**：厂商名称。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

参数	是否必选	参数类型	描述
industry	否	String	**参数说明**：设备所属行业。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
description	否	String	**参数说明**：产品的描述信息。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$！【】’‘“”（）？…~/等字符的组合。

表 1-36 ServiceCapability

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID。注：产品内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
service_type	是	String	**参数说明**：设备的服务类型。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
properties	否	Array of **ServiceProperty** objects	**参数说明**：设备服务支持的属性列表。 **取值范围**：数组长度大小不超过500。
commands	否	Array of **ServiceCommand** objects	**参数说明**：设备服务支持的命令列表。 **取值范围**：数组长度大小不超过500。
events	否	Array of **ServiceEvent** objects	**参数说明**：设备服务支持的事件列表。目前暂未支持自定义事件，创建/修改产品时无需定义该字段。 **取值范围**：数组长度大小不超过500。
description	否	String	**参数说明**：设备服务的描述信息。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$！【】’‘“”（）？…~/等字符的组合。

参数	是否必选	参数类型	描述
option	否	String	**参数说明**：指定设备服务是否必选。目前本字段为非功能性字段，仅起到标识作用。 **取值范围**： ● Master：主服务 ● Mandatory：必选服务 ● Optional：可选服务 默认值为Optional。 缺省值：**Optional**

表 1-37 ServiceProperty

参数	是否必选	参数类型	描述
property_name	是	String	**参数说明**：设备属性名称。注：设备服务内不允许重复。属性名称作为设备影子JSON文档中的key不支持特殊字符：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。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	是	String	**参数说明**：设备属性的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	否	Boolean	**参数说明**：设备属性是否必选。默认为false。 缺省值：**false**
enum_list	否	Array of strings	**参数说明**：设备属性的枚举值列表。
min	否	String	**参数说明**：设备属性的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**

参数	是否必选	参数类型	描述
max	否	String	**参数说明**：设备属性的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：设备属性的最大长度。
step	否	Double	**参数说明**：设备属性的步长。
unit	否	String	**参数说明**：设备属性的单位。 **取值范围**：长度不超过16。 最大长度：**16**
method	是	String	**参数说明**：设备属性的访问模式。 **取值范围**：RWE，RW，RE，WE，E，W，R。 ● R：属性值可读 ● W：属性值可写 ● E：属性值可订阅，即属性值变化时上报事件
description	否	String	**参数说明**：设备属性的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$！【】’‘“”（）？…~/等字符的组合。
default_value	否	Object	**参数说明**：设备属性的默认值。如果设置了默认值，使用该产品创建设备时，会将该属性的默认值写入到该设备的设备影子预期数据中，待设备上线时将该属性默认值下发给设备。

表 1-38 ServiceCommand

参数	是否必选	参数类型	描述
command_name	是	String	**参数说明**：设备命令名称。注：设备服务内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

参数	是否必选	参数类型	描述
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令的参数列表。
responses	否	Array of **ServiceCommandResponse** objects	**参数说明**：设备命令的响应列表。

表 1-39 ServiceCommandResponse

参数	是否必选	参数类型	描述
response_name	是	String	**参数说明**：设备命令响应名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令响应的参数列表。

表 1-40 ServiceEvent

参数	是否必选	参数类型	描述
event_type	是	String	**参数说明**：设备事件类型。注：设备服务内不允许重复。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备事件的参数列表。

表 1-41 ServiceCommandPara

参数	是否必选	参数类型	描述
para_name	是	String	**参数说明**：参数的名称。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

参数	是否必选	参数类型	描述
data_type	是	String	**参数说明**：参数的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	否	Boolean	**参数说明**：参数是否必选。默认为false。 缺省值：**false**
enum_list	否	Array of strings	**参数说明**：参数的枚举值列表。
min	否	String	**参数说明**：参数的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	否	String	**参数说明**：参数的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：参数的最大长度。
step	否	Double	**参数说明**：参数的步长。
unit	否	String	**参数说明**：参数的单位。 **取值范围**：长度不超过16。 最大长度：**16**
description	否	String	**参数说明**：参数的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$! 【 】’ ‘“”（）？…~/等字符的组合。

响应参数

状态码： 200

表 1-42 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID。
app_name	String	资源空间名称。

参数	参数类型	描述
product_id	String	产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。
name	String	产品名称。
device_type	String	设备类型。
protocol_type	String	设备使用的协议类型。取值范围：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF，OPC-UA，OPC-DA，Other。
data_format	String	设备上报数据的格式，取值范围：json，binary。
manufacturer_name	String	厂商名称。
industry	String	设备所属行业。
description	String	产品的描述信息。
service_capabilities	Array of **ServiceCapability** objects	设备的服务能力列表。
create_time	String	在物联网平台创建产品的时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-43 ServiceCapability

参数	参数类型	描述
service_id	String	**参数说明**：设备的服务ID。注：产品内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
service_type	String	**参数说明**：设备的服务类型。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-$等字符的组合。
properties	Array of **ServiceProperty** objects	**参数说明**：设备服务支持的属性列表。 **取值范围**：数组长度大小不超过500。
commands	Array of **ServiceCommand** objects	**参数说明**：设备服务支持的命令列表。 **取值范围**：数组长度大小不超过500。
events	Array of **ServiceEvent** objects	**参数说明**：设备服务支持的事件列表。目前暂未支持自定义事件，创建/修改产品时无需定义该字段。 **取值范围**：数组长度大小不超过500。

参数	参数类型	描述
description	String	**参数说明**：设备服务的描述信息。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$!【】’ ‘“”（）？…~/等字符的组合。
option	String	**参数说明**：指定设备服务是否必选。目前本字段为非功能性字段，仅起到标识作用。 **取值范围**： ● Master：主服务 ● Mandatory：必选服务 ● Optional：可选服务 默认值为Optional。 缺省值：**Optional**

表 1-44 ServiceProperty

参数	参数类型	描述
property_name	String	**参数说明**：设备属性名称。注：设备服务内不允许重复。属性名称作为设备影子JSON文档中的key不支持特殊字符：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。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	String	**参数说明**：设备属性的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	Boolean	**参数说明**：设备属性是否必选。默认为false。 缺省值：**false**
enum_list	Array of strings	**参数说明**：设备属性的枚举值列表。
min	String	**参数说明**：设备属性的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	String	**参数说明**：设备属性的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	Integer	**参数说明**：设备属性的最大长度。
step	Double	**参数说明**：设备属性的步长。

参数	参数类型	描述
unit	String	**参数说明**：设备属性的单位。 **取值范围**：长度不超过16。 最大长度：**16**
method	String	**参数说明**：设备属性的访问模式。 **取值范围**：RWE，RW，RE，WE，E，W，R。 ● R：属性值可读 ● W：属性值可写 ● E：属性值可订阅，即属性值变化时上报事件
description	String	**参数说明**：设备属性的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$!【】’‘“”（）？…~/等字符的组合。
default_value	Object	**参数说明**：设备属性的默认值。如果设置了默认值，使用该产品创建设备时，会将该属性的默认值写入到该设备的设备影子预期数据中，待设备上线时将该属性默认值下发给设备。

表 1-45 ServiceCommand

参数	参数类型	描述
command_name	String	**参数说明**：设备命令名称。注：设备服务内不允许重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备命令的参数列表。
responses	Array of **ServiceCommandResponse** objects	**参数说明**：设备命令的响应列表。

表 1-46 ServiceCommandResponse

参数	参数类型	描述
response_name	String	**参数说明**：设备命令响应名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

参数	参数类型	描述
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备命令响应的参数列表。

表 1-47 ServiceEvent

参数	参数类型	描述
event_type	String	**参数说明**：设备事件类型。注：设备服务内不允许重复。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
paras	Array of **ServiceCommandPara** objects	**参数说明**：设备事件的参数列表。

表 1-48 ServiceCommandPara

参数	参数类型	描述
para_name	String	**参数说明**：参数的名称。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
data_type	String	**参数说明**：参数的数据类型。 **取值范围**：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
required	Boolean	**参数说明**：参数是否必选。默认为false。 缺省值：**false**
enum_list	Array of strings	**参数说明**：参数的枚举值列表。
min	String	**参数说明**：参数的最小值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max	String	**参数说明**：参数的最大值。 **取值范围**：长度1-16。 最小长度：**1** 最大长度：**16**
max_length	Integer	**参数说明**：参数的最大长度。

参数	参数类型	描述
step	Double	**参数说明**：参数的步长。
unit	String	**参数说明**：参数的单位。 **取值范围**：长度不超过16。 最大长度：**16**
description	String	**参数说明**：参数的描述。 **取值范围**：长度不超过128，只允许中文、字母、数字、空白字符、以及_?'#().,;&%@!-，、：；。￥$! 【】’ ‘ “ ”（）？…~/等字符的组合。

请求示例

修改一个产品，将产品名称修改为Thermometer，服务能力修改为temperature。

```
PUT https://{endpoint}/v5/iot/{project_id}/products/{product_id}

{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "name" : "Thermometer，",
  "device_type" : "Thermometer",
  "protocol_type" : "MQTT",
  "data_format" : "json",
  "service_capabilities" : [ {
    "service_id" : "temperature。",
    "service_type" : "temperature",
    "properties" : [ {
      "property_name" : "temperature",
      "data_type" : "decimal",
      "required" : true,
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "centigrade",
      "method" : "RW",
      "description" : "force",
      "default_value" : {
        "color" : "red",
        "size" : 1
      }
    } ],
    "commands" : [ {
      "command_name" : "reboot",
      "paras" : [ {
        "para_name" : "force",
        "data_type" : "string",
        "required" : false,
        "enum_list" : null,
        "min" : "1",
        "max" : "100",
        "max_length" : 100,
        "step" : 0.1,
        "unit" : "km/h",
        "description" : "force"
      } ],
      "responses" : [ {
        "response_name" : "ACK",
        "paras" : [ {
          "para_name" : "force",
```



```
        "data_type" : "string",
        "required" : false,
        "enum_list" : null,
        "min" : "1",
        "max" : "100",
        "max_length" : 100,
        "step" : 0.1,
        "unit" : "km/h",
        "description" : "force"
      } ]
    } ]
  } ],
  "events" : [ {
    "event_type" : "reboot",
    "paras" : [ {
      "para_name" : "force",
      "data_type" : "string",
      "required" : false,
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "km/h",
      "description" : "force"
    } ]
  } ],
  "description" : "temperature",
  "option" : "Mandatory"
 } ],
 "manufacturer_name" : "ABC",
 "industry" : "smartCity",
 "description" : "this is a thermometer produced by Huawei"
}
```

响应示例

状态码： 200

Successful response

```
{
 "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
 "app_name" : "testAPP01",
 "product_id" : "5ba24f5ebbe8f56f5a14f605",
 "name" : "Thermometer",
 "device_type" : "Thermometer",
 "protocol_type" : "MQTT",
 "data_format" : "json",
 "manufacturer_name" : "ABC",
 "industry" : "smartCity",
 "description" : "this is a thermometer produced by Huawei",
 "service_capabilities" : [ {
  "service_id" : "temperature",
  "service_type" : "temperature",
  "properties" : [ {
    "property_name" : "temperature",
    "required" : true,
    "data_type" : "decimal",
    "enum_list" : null,
    "min" : "1",
    "max" : "100",
    "max_length" : 100,
    "step" : 0.1,
    "unit" : "centigrade",
    "method" : "RW",
    "description" : "force",
    "default_value" : {
      "color" : "red",
```



```
      "size" : 1
    }
  } ],
  "commands" : [ {
    "command_name" : "reboot",
    "paras" : [ {
      "para_name" : "force",
      "required" : false,
      "data_type" : "string",
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "km/h",
      "description" : "force"
    } ],
    "responses" : [ {
      "response_name" : "ACK",
      "paras" : [ {
        "para_name" : "force",
        "required" : false,
        "data_type" : "string",
        "enum_list" : null,
        "min" : "1",
        "max" : "100",
        "max_length" : 100,
        "step" : 0.1,
        "unit" : "km/h",
        "description" : "force"
      } ]
    } ]
  } ],
  "events" : [ {
    "event_type" : "reboot",
    "paras" : [ {
      "para_name" : "force",
      "required" : false,
      "data_type" : "string",
      "enum_list" : null,
      "min" : "1",
      "max" : "100",
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "km/h",
      "description" : "force"
    } ]
  } ],
  "description" : "temperature",
  "option" : "Mandatory"
 } ],
 "create_time" : "20190303T081011Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改一个产品，将产品名称修改为Thermometer，服务能力修改为temperature。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
```



```
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateProductSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateProductRequest request = new UpdateProductRequest();
        UpdateProduct body = new UpdateProduct();
        List<ServiceCommandPara> listEventsParas = new ArrayList<>();
        listEventsParas.add(
            new ServiceCommandPara()
                .withParaName("force")
                .withDataType("string")
                .withRequired(false)
                .withMin("1")
                .withMax("100")
                .withMaxLength(100)
                .withStep((double)0.1)
                .withUnit("km/h")
                .withDescription("force")
        );
        List<ServiceEvent> listServiceCapabilitiesEvents = new ArrayList<>();
        listServiceCapabilitiesEvents.add(
            new ServiceEvent()
                .withEventType("reboot")
                .withParas(listEventsParas)
        );
        List<ServiceCommandPara> listResponsesParas = new ArrayList<>();
        listResponsesParas.add(
            new ServiceCommandPara()
                .withParaName("force")
                .withDataType("string")
                .withRequired(false)
                .withMin("1")
                .withMax("100")
                .withMaxLength(100)
                .withStep((double)0.1)
                .withUnit("km/h")
                .withDescription("force")
```

```
);
List<ServiceCommandResponse> listCommandsResponses = new ArrayList<>();
listCommandsResponses.add(
    new ServiceCommandResponse()
        .withResponseName("ACK")
        .withParas(listResponsesParas)
);
List<ServiceCommandPara> listCommandsParas = new ArrayList<>();
listCommandsParas.add(
    new ServiceCommandPara()
        .withParaName("force")
        .withDataType("string")
        .withRequired(false)
        .withMin("1")
        .withMax("100")
        .withMaxLength(100)
        .withStep((double)0.1)
        .withUnit("km/h")
        .withDescription("force")
);
List<ServiceCommand> listServiceCapabilitiesCommands = new ArrayList<>();
listServiceCapabilitiesCommands.add(
    new ServiceCommand()
        .withCommandName("reboot")
        .withParas(listCommandsParas)
        .withResponses(listCommandsResponses)
);
List<ServiceProperty> listServiceCapabilitiesProperties = new ArrayList<>();
listServiceCapabilitiesProperties.add(
    new ServiceProperty()
        .withPropertyName("temperature")
        .withDataType("decimal")
        .withRequired(true)
        .withMin("1")
        .withMax("100")
        .withMaxLength(100)
        .withStep((double)0.1)
        .withUnit("centigrade")
        .withMethod("RW")
        .withDescription("force")
        .withDefaultValue("{\"color\":\"red\",\"size\":1}")
);
List<ServiceCapability> listbodyServiceCapabilities = new ArrayList<>();
listbodyServiceCapabilities.add(
    new ServiceCapability()
        .withServiceId("temperature。")
        .withServiceType("temperature")
        .withProperties(listServiceCapabilitiesProperties)
        .withCommands(listServiceCapabilitiesCommands)
        .withEvents(listServiceCapabilitiesEvents)
        .withDescription("temperature")
        .withOption("Mandatory")
);
body.withDescription("this is a thermometer produced by Huawei");
body.withIndustry("smartCity");
body.withManufacturerName("ABC");
body.withServiceCapabilities(listbodyServiceCapabilities);
body.withDataFormat("json");
body.withProtocolType("MQTT");
body.withDeviceType("Thermometer");
body.withName("Thermometer,");
body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
request.withBody(body);
try {
    UpdateProductResponse response = client.updateProduct(request);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
```



```
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改一个产品，将产品名称修改为Thermometer，服务能力修改为temperature。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateProductRequest()
        listParasEvents = [
            ServiceCommandPara(
                para_name="force",
                data_type="string",
                required=False,
                min="1",
                max="100",
                max_length=100,
                step=0.1,
                unit="km/h",
                description="force"
            )
        ]
        listEventsServiceCapabilities = [
            ServiceEvent(
                event_type="reboot",
                paras=listParasEvents
            )
        ]
        listParasResponses = [
            ServiceCommandPara(
                para_name="force",
```

```
            data_type="string",
            required=False,
            min="1",
            max="100",
            max_length=100,
            step=0.1,
            unit="km/h",
            description="force"
        )
    ]
    listResponsesCommands = [
        ServiceCommandResponse(
            response_name="ACK",
            paras=listParasResponses
        )
    ]
    listParasCommands = [
        ServiceCommandPara(
            para_name="force",
            data_type="string",
            required=False,
            min="1",
            max="100",
            max_length=100,
            step=0.1,
            unit="km/h",
            description="force"
        )
    ]
    listCommandsServiceCapabilities = [
        ServiceCommand(
            command_name="reboot",
            paras=listParasCommands,
            responses=listResponsesCommands
        )
    ]
    listPropertiesServiceCapabilities = [
        ServiceProperty(
            property_name="temperature",
            data_type="decimal",
            required=True,
            min="1",
            max="100",
            max_length=100,
            step=0.1,
            unit="centigrade",
            method="RW",
            description="force",
            default_value="{\"color\":\"red\",\"size\":1}"
        )
    ]
    listServiceCapabilitiesbody = [
        ServiceCapability(
            service_id="temperature。",
            service_type="temperature",
            properties=listPropertiesServiceCapabilities,
            commands=listCommandsServiceCapabilities,
            events=listEventsServiceCapabilities,
            description="temperature",
            option="Mandatory"
        )
    ]
    request.body = UpdateProduct(
        description="this is a thermometer produced by Huawei",
        industry="smartCity",
        manufacturer_name="ABC",
        service_capabilities=listServiceCapabilitiesbody,
        data_format="json",
        protocol_type="MQTT",
```



```
            device_type="Thermometer",
            name="Thermometer,",
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka"
        )
        response = client.update_product(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改一个产品，将产品名称修改为Thermometer，服务能力修改为temperature。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateProductRequest{}
    requiredParas:= false
    minParas:= "1"
    maxParas:= "100"
    maxLengthParas:= int32(100)
    stepParas:= float64(0.1)
    unitParas:= "km/h"
    descriptionParas:= "force"
    var listParasEvents = []model.ServiceCommandPara{
        {
            ParaName: "force",
            DataType: "string",
            Required: &requiredParas,
            Min: &minParas,
            Max: &maxParas,
            MaxLength: &maxLengthParas,
```



```
        Step: &stepParas,
        Unit: &unitParas,
        Description: &descriptionParas,
    },
}
var listEventsServiceCapabilities = []model.ServiceEvent{
    {
        EventType: "reboot",
        Paras: &listParasEvents,
    },
}
requiredParas1:= false
minParas1:= "1"
maxParas1:= "100"
maxLengthParas1:= int32(100)
stepParas1:= float64(0.1)
unitParas1:= "km/h"
descriptionParas1:= "force"
var listParasResponses = []model.ServiceCommandPara{
    {
        ParaName: "force",
        DataType: "string",
        Required: &requiredParas1,
        Min: &minParas1,
        Max: &maxParas1,
        MaxLength: &maxLengthParas1,
        Step: &stepParas1,
        Unit: &unitParas1,
        Description: &descriptionParas1,
    },
}
var listResponsesCommands = []model.ServiceCommandResponse{
    {
        ResponseName: "ACK",
        Paras: &listParasResponses,
    },
}
requiredParas2:= false
minParas2:= "1"
maxParas2:= "100"
maxLengthParas2:= int32(100)
stepParas2:= float64(0.1)
unitParas2:= "km/h"
descriptionParas2:= "force"
var listParasCommands = []model.ServiceCommandPara{
    {
        ParaName: "force",
        DataType: "string",
        Required: &requiredParas2,
        Min: &minParas2,
        Max: &maxParas2,
        MaxLength: &maxLengthParas2,
        Step: &stepParas2,
        Unit: &unitParas2,
        Description: &descriptionParas2,
    },
}
var listCommandsServiceCapabilities = []model.ServiceCommand{
    {
        CommandName: "reboot",
        Paras: &listParasCommands,
        Responses: &listResponsesCommands,
    },
}
requiredProperties:= true
minProperties:= "1"
maxProperties:= "100"
maxLengthProperties:= int32(100)
stepProperties:= float64(0.1)
```



```
    unitProperties:= "centigrade"
    descriptionProperties:= "force"
    defaultValueProperties:= "{\"color\":\"red\",\"size\":1}"
    var defaultValuePropertiesInterface interface{} = defaultValueProperties
    var listPropertiesServiceCapabilities = []model.ServiceProperty{
        {
            PropertyName: "temperature",
            DataType: "decimal",
            Required: &requiredProperties,
            Min: &minProperties,
            Max: &maxProperties,
            MaxLength: &maxLengthProperties,
            Step: &stepProperties,
            Unit: &unitProperties,
            Method: "RW",
            Description: &descriptionProperties,
            DefaultValue: &defaultValuePropertiesInterface,
        },
    }
    descriptionServiceCapabilities:= "temperature"
    optionServiceCapabilities:= "Mandatory"
    var listServiceCapabilitiesbody = []model.ServiceCapability{
        {
            ServiceId: "temperature。",
            ServiceType: "temperature",
            Properties: &listPropertiesServiceCapabilities,
            Commands: &listCommandsServiceCapabilities,
            Events: &listEventsServiceCapabilities,
            Description: &descriptionServiceCapabilities,
            Option: &optionServiceCapabilities,
        },
    }
    descriptionUpdateProduct:= "this is a thermometer produced by Huawei"
    industryUpdateProduct:= "smartCity"
    manufacturerNameUpdateProduct:= "ABC"
    dataFormatUpdateProduct:= "json"
    protocolTypeUpdateProduct:= "MQTT"
    deviceTypeUpdateProduct:= "Thermometer"
    nameUpdateProduct:= "Thermometer,"
    appIdUpdateProduct:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    request.Body = &model.UpdateProduct{
        Description: &descriptionUpdateProduct,
        Industry: &industryUpdateProduct,
        ManufacturerName: &manufacturerNameUpdateProduct,
        ServiceCapabilities: &listServiceCapabilitiesbody,
        DataFormat: &dataFormatUpdateProduct,
        ProtocolType: &protocolTypeUpdateProduct,
        DeviceType: &deviceTypeUpdateProduct,
        Name: &nameUpdateProduct,
        AppId: &appIdUpdateProduct,
    }
    response, err := client.UpdateProduct(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Successful response
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.1.5 删除产品

功能介绍

应用服务器可调用此接口删除已导入物联网平台的指定产品模型。此接口仅删除了产品，未删除关联的插件，在产品下存在设备时，该产品不允许删除。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/products/{product_id}

表 1-49 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
product_id	是	String	**参数说明**：产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

表 1-50 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数，指定要删除的产品属于哪个资源空间；若不携带，则优先删除**默认资源空间**下产品，如默认资源空间下无对应产品，则按照产品创建时间删除最早创建产品。如果用户存在多资源空间，同时又不想携带该参数，可以联系华为技术支持对用户数据做资源空间合并。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-51 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除指定产品。


```
DELETE https://{endpoint}/v5/iot/{project_id}/products/{product_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteProductSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteProductRequest request = new DeleteProductRequest();
        request.withAppId("<app_id>");
        try {
            DeleteProductResponse response = client.deleteProduct(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
```



```
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteProductRequest()
        request.app_id = "<app_id>"
        response = client.delete_product(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"
```

```
    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteProductRequest{}
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    response, err := client.DeleteProduct(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2 设备管理

1.4.2.1 创建设备

功能介绍

应用服务器可调用此接口在物联网平台创建一个设备，仅在创建后设备才可以接入物联网平台。

- 该接口支持使用gateway_id参数指定在父设备下创建一个子设备，并且支持多级子设备，当前最大支持二级子设备。
- 该接口同时还支持对设备进行初始配置，接口会读取创建设备请求参数product_id对应的产品详情，如果产品的属性有定义默认值，则会将该属性默认值写入该设备的设备影子中。
- 用户还可以使用创建设备请求参数shadow字段为设备指定初始配置，指定后将会根据service_id和desired设置的属性值与产品中对应属性的默认值比对，如果不同，则将以shadow字段中设置的属性值为准写入到设备影子中。
- 该接口仅支持创建单个设备，如需批量注册设备，请参见 **创建批量任务**。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices

表 1-52 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-53 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。

参数	是否必选	参数类型	描述
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-54 请求 Body 参数

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：设备ID，全局唯一，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。 **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，建议不少于4个字符。
node_id	是	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为node_id。 设备标识码长度为1到64个字符，包含英文字母、数字、连接号-和下划线_。注意：NB设备由于模组烧录信息后无法配置，所以NB设备会校验node_id全局唯一。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合，建议不少于4个字符。
device_name	否	String	**参数说明**：设备名称，资源空间下唯一，用于资源空间下唯一标识一个设备。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合，建议不少于4个字符。 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
product_id	是	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
auth_info	否	**AuthInfo** object	**参数说明**：设备的接入认证信息。
description	否	String	**参数说明**：设备的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
gateway_id	否	String	**参数说明**：网关ID，用于标识设备所属的父设备，即父设备的设备ID。携带该参数时，表示在该父设备下创建一个子设备，这个子设备不与平台直连，此时必须保证这个父设备在平台已存在，创建成功后子设备的gateway_id等于该参数值；不携带该参数时，表示创建一个和平台直连的设备，创建成功后设备的device_id和gateway_id一致。注意：当前平台最多支持二级子设备。 **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。
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的设备归属到哪个资源空间下，否则创建的设备将会归属到**默认资源空间**下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
extension_info	否	Object	**参数说明**：设备扩展信息。用户可以自定义任何想要的扩展信息，如果在创建设备时为子设备指定该字段，将会通过MQTT接口“平台通知网关子设备新增“将该信息通知给网关。字段值大小上限为1K。

参数	是否必选	参数类型	描述
shadow	否	Array of InitialDesired objects	**参数说明**：设备初始配置。用户使用该字段可以为设备指定初始配置，指定后将会根据service_id和desired设置的属性值与产品中对应属性的默认值比对，如果不同，则将以shadow字段中设置的属性值为准写入到设备影子中。service_id的值和desired内的属性必须是profile中定义的。

表 1-55 AuthInfo

参数	是否必选	参数类型	描述
auth_type	否	String	**参数说明**：鉴权类型。注意：不填写auth_type默认为密钥认证接入方式(SECRET)。 **取值范围**： • SECRET:使用密钥认证接入方式。 • CERTIFICATES:使用证书认证接入方式。
secret	否	String	**参数说明**：设备密钥，认证类型使用密钥认证接入(SECRET)可填写该字段。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入；查询设备列表接口不返回该参数。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**
fingerprint	否	String	**参数说明**：证书指纹，认证类型使用证书认证接入(CERTIFICATES)可填写该字段，注册设备时不填写该字段则取第一次设备接入时的证书指纹。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。

参数	是否必选	参数类型	描述
secure_access	否	Boolean	**参数说明**：指设备是否通过安全协议方式接入。 **取值范围**： • true：通过安全协议方式接入。 • false：通过非安全协议方式接入。非安全接入的设备存在被仿冒等安全风险，请谨慎使用。 缺省值：**true**
timeout	否	Integer	**参数说明**：设备接入的有效时间，单位：秒，默认值：0 若设备在有效时间内未接入物联网平台并激活，则平台会删除该设备的注册信息。若设置为“0”，则表示平台不会删除该设备的注册信息（建议填写为“0”）。注意：该参数只对直连设备生效。 最小值：**0** 最大值：**2147483647** 缺省值：**0**

表 1-56 InitialDesired

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID，在设备关联的产品模型中定义。 **取值范围**：长度不超过32，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
desired	是	Object	**参数说明**：设备初始配置属性数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)，目前如样例所示只支持一层结构；这里设置的属性值与产品中对应属性的默认值比对，如果不同，则将以该字段中设置的属性值为准写入到设备影子中；如果想要删除整个desired可以填写空object(例如"desired":{})，如果想要删除某一个属性期望值可以将属性置位null(例如{"temperature":null})

响应参数

状态码： 201

表 1-57 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID。 最大长度：**36**
app_name	String	资源空间名称。
device_id	String	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。
node_id	String	设备标识码，通常使用IMEI、MAC地址或Serial No作为node_id。
gateway_id	String	网关ID，用于标识设备所属的父设备，即父设备的设备ID。当设备是直连设备时，gateway_id与设备的device_id一致。当设备是非直连设备时，gateway_id为设备所关联的父设备的device_id。
device_name	String	设备名称。
node_type	String	设备节点类型。 ● ENDPOINT：非直连设备。 ● GATEWAY：直连设备或网关。 ● UNKNOWN：未知。
description	String	设备的描述信息。
fw_version	String	设备的固件版本。
sw_version	String	设备的软件版本。
device_sdk_version	String	设备的sdk信息。
auth_info	**AuthInfoRes** object	设备的接入认证信息。
product_id	String	设备关联的产品ID，用于唯一标识一个产品模型。
product_name	String	设备关联的产品名称。

参数	参数类型	描述
status	String	设备的状态。 ● ONLINE：设备在线。 ● OFFLINE：设备离线。 ● ABNORMAL：设备异常。 ● INACTIVE：设备未激活。 ● FROZEN：设备冻结。
create_time	String	在物联网平台注册设备的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
connection_status_update_time	String	设备最近一次连接状态(ONLINE:在线，OFFLINE：离线，ABNORMAL：异常)变化时间。格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' ，如2015-12-12T12:12:122Z。
active_time	String	设备激活时间。格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' ，如2015-12-12T12:12:122Z。
tags	Array of TagV5DTO objects	设备的标签列表。
extension_info	Object	设备扩展信息。用户可以自定义任何想要的扩展信息，如果在创建设备时为子设备指定该字段，将会通过MQTT接口“平台通知网关子设备新增“将该信息通知给网关。

表 1-58 AuthInfoRes

参数	参数类型	描述
auth_type	String	**参数说明**：鉴权类型。注意：不填写auth_type默认为密钥认证接入方式(SECRET)。 **取值范围**： ● SECRET:使用密钥认证接入方式。 ● CERTIFICATES:使用证书认证接入方式。
secret	String	**参数说明**：设备密钥，认证类型使用密钥认证接入(SECRET)可填写该字段。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入；查询设备列表接口不返回该参数。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**

参数	参数类型	描述
secondary_secret	String	**参数说明**：设备备用密钥，认证类型使用密钥认证接入(SECRET)该字段有效，当主密钥校验不通过时，会启用辅密钥校验，辅密钥与主密钥有相同的效力；辅密钥对coap协议接入的设备不生效。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入；查询设备列表接口不返回该参数。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**
fingerprint	String	**参数说明**：证书指纹，认证类型使用证书认证接入(CERTIFICATES)该字段有效，注册设备时不填写该字段则取第一次设备接入时的证书指纹。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。
secondary_fingerprint	String	**参数说明**：证书备用指纹，认证类型使用证书认证接入(CERTIFICATES)该字段有效，当主指纹校验不通过时，会启用辅指纹校验，辅指纹与主指纹有相同的效力。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。
secure_access	Boolean	**参数说明**：指设备是否通过安全协议方式接入。 **取值范围**： ● true：通过安全协议方式接入。 ● false：通过非安全协议方式接入。非安全接入的设备存在被仿冒等安全风险，请谨慎使用。 缺省值：**true**
timeout	Integer	**参数说明**：设备接入的有效时间，单位：秒，默认值：0 若设备在有效时间内未接入物联网平台并激活，则平台会删除该设备的注册信息。若设置为“0”，则表示平台不会删除该设备的注册信息（建议填写为“0”）。 注意：该参数只对直连设备生效。 最小值：**0** 最大值：**2147483647** 缺省值：**0**

表 1-59 TagV5DTO

参数	参数类型	描述
tag_key	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。
tag_value	String	**参数说明**：标签值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。

请求示例

- 创建设备，认证类型为密钥认证。

```
POST https://{endpoint}/v5/iot/{project_id}/devices

{
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "node_id" : "ABC123456789",
  "device_name" : "dianadevice",
  "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secret" : "3b935a250c50dc2c6d481d048cefdc3c",
    "secure_access" : true
  },
  "description" : "watermeter device",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  },
  "shadow" : [ {
    "service_id" : "WaterMeter",
    "desired" : {
      "temperature" : "60"
    }
  } ]
}
```

- 创建设备，认证类型为证书认证。

```
POST https://{endpoint}/v5/iot/{project_id}/devices

{
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "node_id" : "ABC123456789",
  "device_name" : "dianadevice",
  "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
  "auth_info" : {
    "auth_type" : "CERTIFICATES",
    "fingerprint" : "dc0f1016f495157344ac5f1296335cff725ef22f",
    "secure_access" : true
  },
  "description" : "watermeter device",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  },
  "shadow" : [ {
```



```
    "service_id" : "WaterMeter",
    "desired" : {
      "temperature" : "60"
    }
  } ]
}
```

- 创建设备，设备类型非直连设备。

```
POST https://{endpoint}/v5/iot/{project_id}/devices

{
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "node_id" : "ABC123456789",
  "device_name" : "dianadevice",
  "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
  "description" : "watermeter device",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "gateway_id" : "5a332c1a-d14f-489c-a8e7-4753b1bb0872",
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  },
  "shadow" : [ {
    "service_id" : "WaterMeter",
    "desired" : {
      "temperature" : "60"
    }
  } ]
}
```

响应示例

状态码： 201

Created

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "app_name" : "testAPP01",
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "node_id" : "ABC123456789",
  "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "device_name" : "dianadevice",
  "node_type" : "ENDPOINT",
  "description" : "watermeter device",
  "fw_version" : "1.1.0",
  "sw_version" : "1.1.0",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secret" : "3b93****fdc3c",
    "fingerprint" : "dc0f****f22f",
    "secure_access" : true,
    "timeout" : 0
  },
  "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
  "product_name" : "Thermometer",
  "status" : "INACTIVE",
  "create_time" : "20190303T081011Z",
  "connection_status_update_time" : null,
  "active_time" : null,
  "tags" : [ {
    "tag_key" : "testTagName",
    "tag_value" : "testTagValue"
  } ],
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  }
}
```


SDK 代码示例

SDK代码示例如下。

Java

- 创建设备，认证类型为密钥认证。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class AddDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddDeviceRequest request = new AddDeviceRequest();
        AddDevice body = new AddDevice();
        List<InitialDesired> listbodyShadow = new ArrayList<>();
        listbodyShadow.add(
            new InitialDesired()
                .withServiceId("WaterMeter")
                .withDesired("{\"temperature\":\"60\"}")
        );
        AuthInfo authInfobody = new AuthInfo();
        authInfobody.withAuthType("SECRET")
            .withSecret("3b935a250c50dc2c6d481d048cefdc3c")
            .withSecureAccess(true);
        body.withShadow(listbodyShadow);
        body.withExtensionInfo("{\"aaa\":\"xxx\",\"bbb\":0}");
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withDescription("watermeter device");
        body.withAuthInfo(authInfobody);
        body.withProductId("b640f4c203b7910fc3cbd446ed437cbd");
        body.withDeviceName("dianadevice");
```



```
        body.withNodeId("ABC123456789");
        body.withDeviceId("d4922d8a-6c8e-4396-852c-164aefa6638f");
        request.withBody(body);
        try {
            AddDeviceResponse response = client.addDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建设备，认证类型为证书认证。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class AddDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddDeviceRequest request = new AddDeviceRequest();
        AddDevice body = new AddDevice();
        List<InitialDesired> listbodyShadow = new ArrayList<>();
        listbodyShadow.add(
            new InitialDesired()
                .withServiceId("WaterMeter")
                .withDesired("{\"temperature\":\"60\"}")
```

```
        );
        AuthInfo authInfobody = new AuthInfo();
        authInfobody.withAuthType("CERTIFICATES")
            .withFingerprint("dc0f1016f495157344ac5f1296335cff725ef22f")
            .withSecureAccess(true);
        body.withShadow(listbodyShadow);
        body.withExtensionInfo("{\"aaa\":\"xxx\",\"bbb\":0}");
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withDescription("watermeter device");
        body.withAuthInfo(authInfobody);
        body.withProductId("b640f4c203b7910fc3cbd446ed437cbd");
        body.withDeviceName("dianadevice");
        body.withNodeId("ABC123456789");
        body.withDeviceId("d4922d8a-6c8e-4396-852c-164aefa6638f");
        request.withBody(body);
        try {
            AddDeviceResponse response = client.addDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建设备，设备类型非直连设备。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class AddDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
```

```
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddDeviceRequest request = new AddDeviceRequest();
        AddDevice body = new AddDevice();
        List<InitialDesired> listbodyShadow = new ArrayList<>();
        listbodyShadow.add(
            new InitialDesired()
                .withServiceId("WaterMeter")
                .withDesired("{\"temperature\":\"60\"}")
        );
        body.withShadow(listbodyShadow);
        body.withExtensionInfo("{\"aaa\":\"xxx\",\"bbb\":0}");
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withGatewayId("5a332c1a-d14f-489c-a8e7-4753b1bb0872");
        body.withDescription("watermeter device");
        body.withProductId("b640f4c203b7910fc3cbd446ed437cbd");
        body.withDeviceName("dianadevice");
        body.withNodeId("ABC123456789");
        body.withDeviceId("d4922d8a-6c8e-4396-852c-164aefa6638f");
        request.withBody(body);
        try {
            AddDeviceResponse response = client.addDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建设备，认证类型为密钥认证。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
```

```
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddDeviceRequest()
        listShadowbody = [
            InitialDesired(
                service_id="WaterMeter",
                desired="{\"temperature\":\"60\"}"
            )
        ]
        authInfobody = AuthInfo(
            auth_type="SECRET",
            secret="3b935a250c50dc2c6d481d048cefdc3c",
            secure_access=True
        )
        request.body = AddDevice(
            shadow=listShadowbody,
            extension_info="{\"aaa\":\"xxx\",\"bbb\":0}",
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            description="watermeter device",
            auth_info=authInfobody,
            product_id="b640f4c203b7910fc3cbd446ed437cbd",
            device_name="dianadevice",
            node_id="ABC123456789",
            device_id="d4922d8a-6c8e-4396-852c-164aefa6638f"
        )
        response = client.add_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建设备，认证类型为证书认证。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
```



```
        request = AddDeviceRequest()
        listShadowbody = [
            InitialDesired(
                service_id="WaterMeter",
                desired="{\"temperature\":\"60\"}"
            )
        ]
        authInfobody = AuthInfo(
            auth_type="CERTIFICATES",
            fingerprint="dc0f1016f495157344ac5f1296335cff725ef22f",
            secure_access=True
        )
        request.body = AddDevice(
            shadow=listShadowbody,
            extension_info="{\"aaa\":\"xxx\",\"bbb\":0}",
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            description="watermeter device",
            auth_info=authInfobody,
            product_id="b640f4c203b7910fc3cbd446ed437cbd",
            device_name="dianadevice",
            node_id="ABC123456789",
            device_id="d4922d8a-6c8e-4396-852c-164aefa6638f"
        )
        response = client.add_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建设备，设备类型非直连设备。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddDeviceRequest()
        listShadowbody = [
            InitialDesired(
                service_id="WaterMeter",
                desired="{\"temperature\":\"60\"}"
            )
```

```
        ]
        request.body = AddDevice(
            shadow=listShadowbody,
            extension_info="{\"aaa\":\"xxx\",\"bbb\":0}",
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            gateway_id="5a332c1a-d14f-489c-a8e7-4753b1bb0872",
            description="watermeter device",
            product_id="b640f4c203b7910fc3cbd446ed437cbd",
            device_name="dianadevice",
            node_id="ABC123456789",
            device_id="d4922d8a-6c8e-4396-852c-164aefa6638f"
        )
        response = client.add_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建设备，认证类型为密钥认证。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddDeviceRequest{}
    desiredShadow:= "{\"temperature\":\"60\"}"
    var desiredShadowInterface interface{} = desiredShadow
    var listShadowbody = []model.InitialDesired{
        {
            ServiceId: "WaterMeter",
            Desired: &desiredShadowInterface,
        },
```



```
    }
    authTypeAuthInfo:= "SECRET"
    secretAuthInfo:= "3b935a250c50dc2c6d481d048cefdc3c"
    secureAccessAuthInfo:= true
    authInfobody := &model.AuthInfo{
        AuthType: &authTypeAuthInfo,
        Secret: &secretAuthInfo,
        SecureAccess: &secureAccessAuthInfo,
    }
    var extensionInfoAddDevice interface{} = "{\"aaa\":\"xxx\",\"bbb\":0}"
    appIdAddDevice:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    descriptionAddDevice:= "watermeter device"
    deviceNameAddDevice:= "dianadevice"
    deviceIdAddDevice:= "d4922d8a-6c8e-4396-852c-164aefa6638f"
    request.Body = &model.AddDevice{
        Shadow: &listShadowbody,
        ExtensionInfo: &extensionInfoAddDevice,
        AppId: &appIdAddDevice,
        Description: &descriptionAddDevice,
        AuthInfo: authInfobody,
        ProductId: "b640f4c203b7910fc3cbd446ed437cbd",
        DeviceName: &deviceNameAddDevice,
        NodeId: "ABC123456789",
        DeviceId: &deviceIdAddDevice,
    }
    response, err := client.AddDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建设备，认证类型为证书认证。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
```

```
        Build())

    request := &model.AddDeviceRequest{}
    desiredShadow:= "{\"temperature\":\"60\"}"
    var desiredShadowInterface interface{} = desiredShadow
    var listShadowbody = []model.InitialDesired{
        {
            ServiceId: "WaterMeter",
            Desired: &desiredShadowInterface,
        },
    }
    authTypeAuthInfo:= "CERTIFICATES"
    fingerprintAuthInfo:= "dc0f1016f495157344ac5f1296335cff725ef22f"
    secureAccessAuthInfo:= true
    authInfobody := &model.AuthInfo{
        AuthType: &authTypeAuthInfo,
        Fingerprint: &fingerprintAuthInfo,
        SecureAccess: &secureAccessAuthInfo,
    }
    var extensionInfoAddDevice interface{} = "{\"aaa\":\"xxx\",\"bbb\":0}"
    appIdAddDevice:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    descriptionAddDevice:= "watermeter device"
    deviceNameAddDevice:= "dianadevice"
    deviceIdAddDevice:= "d4922d8a-6c8e-4396-852c-164aefa6638f"
    request.Body = &model.AddDevice{
        Shadow: &listShadowbody,
        ExtensionInfo: &extensionInfoAddDevice,
        AppId: &appIdAddDevice,
        Description: &descriptionAddDevice,
        AuthInfo: authInfobody,
        ProductId: "b640f4c203b7910fc3cbd446ed437cbd",
        DeviceName: &deviceNameAddDevice,
        NodeId: "ABC123456789",
        DeviceId: &deviceIdAddDevice,
    }
    response, err := client.AddDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建设备，设备类型非直连设备。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```



```
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddDeviceRequest{}
    desiredShadow:= "{\"temperature\":\"60\"}"
    var desiredShadowInterface interface{} = desiredShadow
    var listShadowbody = []model.InitialDesired{
        {
            ServiceId: "WaterMeter",
            Desired: &desiredShadowInterface,
        },
    }
    var extensionInfoAddDevice interface{} = "{\"aaa\":\"xxx\",\"bbb\":0}"
    appIdAddDevice:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    gatewayIdAddDevice:= "5a332c1a-d14f-489c-a8e7-4753b1bb0872"
    descriptionAddDevice:= "watermeter device"
    deviceNameAddDevice:= "dianadevice"
    deviceIdAddDevice:= "d4922d8a-6c8e-4396-852c-164aefa6638f"
    request.Body = &model.AddDevice{
        Shadow: &listShadowbody,
        ExtensionInfo: &extensionInfoAddDevice,
        AppId: &appIdAddDevice,
        GatewayId: &gatewayIdAddDevice,
        Description: &descriptionAddDevice,
        ProductId: "b640f4c203b7910fc3cbd446ed437cbd",
        DeviceName: &deviceNameAddDevice,
        NodeId: "ABC123456789",
        DeviceId: &deviceIdAddDevice,
    }
    response, err := client.AddDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found

状态码	描述
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.2 查询设备列表

功能介绍

应用服务器可调用此接口查询物联网平台中的设备信息列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/devices

表 1-60 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-61 Query 参数

参数	是否必选	参数类型	描述
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

参数	是否必选	参数类型	描述
gateway_id	否	String	**参数说明**：网关ID，用于标识设备所属的父设备，即父设备的设备ID。携带该参数时，表示查询该设备下的子设备，默认查询下一级子设备，如果需要查询该设备下所有各级子设备，请同时携带is_cascade_query参数为true；不携带该参数时，表示查询用户下所有设备。 **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。
is_cascade_query	否	Boolean	**参数说明**：是否级联查询，该参数仅在同时携带gateway_id时生效。默认值为false。 **取值范围**： ● true：表示查询设备ID等于gateway_id参数的设备下的所有各级子设备。 ● false：表示查询设备ID等于gateway_id参数的设备下的一级子设备。 缺省值：**false**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为node_id。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合。
device_name	否	String	**参数说明**：设备名称，资源空间下唯一，用于资源空间下唯一标识一个设备。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**
start_time	否	String	**参数说明**：查询设备注册时间在startTime之后的记录，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
end_time	否	String	**参数说明**：查询设备注册时间在endTime之前的记录，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的设备列表，不携带该参数则会查询该用户下所有设备列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-62 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-63 响应 Body 参数

参数	参数类型	描述
devices	Array of **QueryDeviceSimplify** objects	设备信息列表。
page	**Page** object	查询结果的分页信息。

表 1-64 QueryDeviceSimplify

参数	参数类型	描述
app_id	String	资源空间ID。
app_name	String	资源空间名称。
device_id	String	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。
node_id	String	设备标识码，通常使用IMEI、MAC地址或Serial No作为node_id。
gateway_id	String	网关ID，用于标识设备所属的父设备，即父设备的设备ID。当设备是直连设备时，gateway_id与设备的device_id一致。当设备是非直连设备时，gateway_id为设备所关联的父设备的device_id。
device_name	String	设备名称。
node_type	String	设备节点类型。 • ENDPOINT：非直连设备。 • GATEWAY：直连设备或网关。 • UNKNOWN：未知。
description	String	设备的描述信息。
fw_version	String	设备的固件版本。
sw_version	String	设备的软件版本。
device_sdk_version	String	设备的sdk信息。
product_id	String	设备关联的产品ID，用于唯一标识一个产品模型。
product_name	String	设备关联的产品名称。
status	String	设备的状态。 • ONLINE：设备在线。 • OFFLINE：设备离线。 • ABNORMAL：设备异常。 • INACTIVE：设备未激活。 • FROZEN：设备冻结。
tags	Array of **TagV5DTO** objects	设备的标签列表。

表 1-65 TagV5DTO

参数	参数类型	描述
tag_key	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。
tag_value	String	**参数说明**：标签值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。

表 1-66 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

列表查询所有设备

```
GET https://{endpoint}/v5/iot/{project_id}/devices
```

响应示例

状态码： 200

OK

```
{
  "devices" : [ {
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "app_name" : "testAPP01",
    "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "node_id" : "ABC123456789",
    "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "device_name" : "dianadevice",
    "node_type" : "ENDPOINT",
    "description" : "watermeter device",
    "fw_version" : "1.1.0",
    "sw_version" : "1.1.0",
    "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
    "product_name" : "Thermometer",
    "status" : "INACTIVE",
    "tags" : [ {
      "tag_key" : "testTagName",
      "tag_value" : "testTagValue"
    } ]
  } ],
  "page" : {
    "count" : 100,
    "marker" : "5c8f3d2d3df1f10d803adbda"
```



```
    }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListDevicesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListDevicesRequest request = new ListDevicesRequest();
        request.withProductId("<product_id>");
        request.withGatewayId("<gateway_id>");
        request.withIsCascadeQuery(<is_cascade_query>);
        request.withNodeId("<node_id>");
        request.withDeviceName("<device_name>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        request.withStartTime("<start_time>");
        request.withEndTime("<end_time>");
        request.withAppId("<app_id>");
        try {
            ListDevicesResponse response = client.listDevices(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
```

```
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListDevicesRequest()
        request.product_id = "<product_id>"
        request.gateway_id = "<gateway_id>"
        request.is_cascade_query = <IsCascadeQuery>
        request.node_id = "<node_id>"
        request.device_name = "<device_name>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        request.start_time = "<start_time>"
        request.end_time = "<end_time>"
        request.app_id = "<app_id>"
        response = client.list_devices(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
```

```
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListDevicesRequest{}
    productIdRequest:= "<product_id>"
    request.ProductId = &productIdRequest
    gatewayIdRequest:= "<gateway_id>"
    request.GatewayId = &gatewayIdRequest
    isCascadeQueryRequest:= <is_cascade_query>
    request.IsCascadeQuery = &isCascadeQueryRequest
    nodeIdRequest:= "<node_id>"
    request.NodeId = &nodeIdRequest
    deviceNameRequest:= "<device_name>"
    request.DeviceName = &deviceNameRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    startTimeRequest:= "<start_time>"
    request.StartTime = &startTimeRequest
    endTimeRequest:= "<end_time>"
    request.EndTime = &endTimeRequest
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    response, err := client.ListDevices(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.3 查询设备

功能介绍

应用服务器可调用此接口查询物联网平台中指定设备的详细信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/devices/{device_id}

表 1-67 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。

请求参数

表 1-68 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-69 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID。 最大长度：**36**
app_name	String	资源空间名称。
device_id	String	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。
node_id	String	设备标识码，通常使用IMEI、MAC地址或Serial No作为node_id。
gateway_id	String	网关ID，用于标识设备所属的父设备，即父设备的设备ID。当设备是直连设备时，gateway_id与设备的device_id一致。当设备是非直连设备时，gateway_id为设备所关联的父设备的device_id。
device_name	String	设备名称。

参数	参数类型	描述
node_type	String	设备节点类型。 • ENDPOINT：非直连设备。 • GATEWAY：直连设备或网关。 • UNKNOWN：未知。
description	String	设备的描述信息。
fw_version	String	设备的固件版本。
sw_version	String	设备的软件版本。
device_sdk_version	String	设备的sdk信息。
auth_info	**AuthInfoRes** object	设备的接入认证信息。
product_id	String	设备关联的产品ID，用于唯一标识一个产品模型。
product_name	String	设备关联的产品名称。
status	String	设备的状态。 • ONLINE：设备在线。 • OFFLINE：设备离线。 • ABNORMAL：设备异常。 • INACTIVE：设备未激活。 • FROZEN：设备冻结。
create_time	String	在物联网平台注册设备的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
connection_status_update_time	String	设备最近一次连接状态(ONLINE:在线，OFFLINE：离线，ABNORMAL：异常)变化时间。格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'，如2015-12-12T12:12:122Z。
active_time	String	设备激活时间。格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'，如2015-12-12T12:12:122Z。
tags	Array of **TagV5DTO** objects	设备的标签列表。
extension_info	Object	设备扩展信息。用户可以自定义任何想要的扩展信息，如果在创建设备时为子设备指定该字段，将会通过MQTT接口“平台通知网关子设备新增“将该信息通知给网关。

表 1-70 AuthInfoRes

参数	参数类型	描述
auth_type	String	**参数说明**：鉴权类型。注意：不填写auth_type默认为密钥认证接入方式(SECRET)。 **取值范围**： ● SECRET:使用密钥认证接入方式。 ● CERTIFICATES:使用证书认证接入方式。
secret	String	**参数说明**：设备密钥，认证类型使用密钥认证接入(SECRET)可填写该字段。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入；查询设备列表接口不返回该参数。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**
secondary_secret	String	**参数说明**：设备备用密钥，认证类型使用密钥认证接入(SECRET)该字段有效，当主密钥校验不通过时，会启用辅密钥校验，辅密钥与主密钥有相同的效力；辅密钥对coap协议接入的设备不生效。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入；查询设备列表接口不返回该参数。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**
fingerprint	String	**参数说明**：证书指纹，认证类型使用证书认证接入(CERTIFICATES)该字段有效，注册设备时不填写该字段则取第一次设备接入时的证书指纹。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。
secondary_fingerprint	String	**参数说明**：证书备用指纹，认证类型使用证书认证接入(CERTIFICATES)该字段有效，当主指纹校验不通过时，会启用辅指纹校验，辅指纹与主指纹有相同的效力。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。
secure_access	Boolean	**参数说明**：指设备是否通过安全协议方式接入。 **取值范围**： ● true：通过安全协议方式接入。 ● false：通过非安全协议方式接入。非安全接入的设备存在被仿冒等安全风险，请谨慎使用。 缺省值：**true**

参数	参数类型	描述
timeout	Integer	**参数说明**：设备接入的有效时间，单位：秒，默认值：0 若设备在有效时间内未接入物联网平台并激活，则平台会删除该设备的注册信息。若设置为“0”，则表示平台不会删除该设备的注册信息（建议填写为“0”）。 注意：该参数只对直连设备生效。 最小值：**0** 最大值：**2147483647** 缺省值：**0**

表 1-71 TagV5DTO

参数	参数类型	描述
tag_key	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。
tag_value	String	**参数说明**：标签值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。

请求示例

查询指定设备详情。

```
GET https://{endpoint}/v5/iot/{project_id}/devices/{device_id}
```

响应示例

状态码： 200

OK

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "app_name" : "testAPP01",
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "node_id" : "ABC123456789",
  "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "device_name" : "dianadevice",
  "node_type" : "ENDPOINT",
  "description" : "watermeter device",
  "fw_version" : "1.1.0",
  "sw_version" : "1.1.0",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secret" : "3b935a250c50dc2c6d481d048cefdc3c",
    "fingerprint" : "dc0f1016f495157344ac5f1296335cff725ef22f",
    "secure_access" : true,
```



```
    "timeout" : 0
  },
  "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
  "product_name" : "Thermometer",
  "status" : "ONLINE",
  "create_time" : "20190303T081011Z",
  "connection_status_update_time" : "2019-03-03T08:10:111Z",
  "active_time" : "2019-03-03T08:10:111Z",
  "tags" : [ {
    "tag_key" : "testTagName",
    "tag_value" : "testTagValue"
  } ],
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDeviceRequest request = new ShowDeviceRequest();
        try {
            ShowDeviceResponse response = client.showDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
```

```
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDeviceRequest()
        response = client.show_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
```

```
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDeviceRequest{}
    response, err := client.ShowDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.4 修改设备

功能介绍

应用服务器可调用此接口修改物联网平台中指定设备的基本信息。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/devices/{device_id}

表 1-72 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

请求参数

表 1-73 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-74 请求 Body 参数

参数	是否必选	参数类型	描述
device_name	否	String	**参数说明**：设备名称，资源空间下唯一，用于资源空间下唯一标识一个设备。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合，建议不少于4个字符。 最小长度：**1** 最大长度：**256**
description	否	String	**参数说明**：设备的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
extension_info	否	Object	**参数说明**：设备扩展信息。用户可以自定义任何想要的扩展信息，修改子设备信息时不会下发给网关。
auth_info	否	**AuthInfoWithoutSecret** object	**参数说明**：设备的接入认证信息。

表 1-75 AuthInfoWithoutSecret

参数	是否必选	参数类型	描述
secure_access	否	Boolean	**参数说明**：指设备是否通过安全协议方式接入。 **取值范围**： • true：通过安全协议方式接入。 • false：通过非安全协议方式接入。非安全接入的设备存在被仿冒等安全风险，请谨慎使用。 缺省值：**true**

参数	是否必选	参数类型	描述
timeout	否	Integer	**参数说明**：设备接入的有效时间，单位：秒，默认值：0。若设备在有效时间内未接入物联网平台并激活，则平台会删除该设备的注册信息。若设置为“0”，则表示平台不会删除该设备的注册信息（建议填写为“0”）。 注意：该参数只对直连设备生效 最小值：**0** 最大值：**2147483647** 缺省值：**0**

响应参数

状态码： 200

表 1-76 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID。 最大长度：**36**
app_name	String	资源空间名称。
device_id	String	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。
node_id	String	设备标识码，通常使用IMEI、MAC地址或Serial No作为node_id。
gateway_id	String	网关ID，用于标识设备所属的父设备，即父设备的设备ID。当设备是直连设备时，gateway_id与设备的device_id一致。当设备是非直连设备时，gateway_id为设备所关联的父设备的device_id。
device_name	String	设备名称。
node_type	String	设备节点类型。 ● ENDPOINT：非直连设备。 ● GATEWAY：直连设备或网关。 ● UNKNOWN：未知。
description	String	设备的描述信息。
fw_version	String	设备的固件版本。

参数	参数类型	描述
sw_version	String	设备的软件版本。
device_sdk_version	String	设备的sdk信息。
auth_info	**AuthInfoRes** object	设备的接入认证信息。
product_id	String	设备关联的产品ID，用于唯一标识一个产品模型。
product_name	String	设备关联的产品名称。
status	String	设备的状态。 ● ONLINE：设备在线。 ● OFFLINE：设备离线。 ● ABNORMAL：设备异常。 ● INACTIVE：设备未激活。 ● FROZEN：设备冻结。
create_time	String	在物联网平台注册设备的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
connection_status_update_time	String	设备最近一次连接状态(ONLINE:在线，OFFLINE：离线，ABNORMAL：异常)变化时间。格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' ，如2015-12-12T12:12:122Z。
active_time	String	设备激活时间。格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' ，如2015-12-12T12:12:122Z。
tags	Array of **TagV5DTO** objects	设备的标签列表。
extension_info	Object	设备扩展信息。用户可以自定义任何想要的扩展信息，如果在创建设备时为子设备指定该字段，将会通过MQTT接口“平台通知网关子设备新增“将该信息通知给网关。

表 1-77 AuthInfoRes

参数	参数类型	描述
auth_type	String	**参数说明**：鉴权类型。注意：不填写auth_type默认为密钥认证接入方式(SECRET)。 **取值范围**： ● SECRET:使用密钥认证接入方式。 ● CERTIFICATES:使用证书认证接入方式。
secret	String	**参数说明**：设备密钥，认证类型使用密钥认证接入(SECRET)可填写该字段。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入；查询设备列表接口不返回该参数。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**
secondary_secret	String	**参数说明**：设备备用密钥，认证类型使用密钥认证接入(SECRET)该字段有效，当主密钥校验不通过时，会启用辅密钥校验，辅密钥与主密钥有相同的效力；辅密钥对coap协议接入的设备不生效。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入；查询设备列表接口不返回该参数。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**
fingerprint	String	**参数说明**：证书指纹，认证类型使用证书认证接入(CERTIFICATES)该字段有效，注册设备时不填写该字段则取第一次设备接入时的证书指纹。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。
secondary_fingerprint	String	**参数说明**：证书备用指纹，认证类型使用证书认证接入(CERTIFICATES)该字段有效，当主指纹校验不通过时，会启用辅指纹校验，辅指纹与主指纹有相同的效力。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。
secure_access	Boolean	**参数说明**：指设备是否通过安全协议方式接入。 **取值范围**： ● true：通过安全协议方式接入。 ● false：通过非安全协议方式接入。非安全接入的设备存在被仿冒等安全风险，请谨慎使用。 缺省值：**true**

参数	参数类型	描述
timeout	Integer	**参数说明**：设备接入的有效时间，单位：秒，默认值：0 若设备在有效时间内未接入物联网平台并激活，则平台会删除该设备的注册信息。若设置为“0”，则表示平台不会删除该设备的注册信息（建议填写为“0”）。 注意：该参数只对直连设备生效。 最小值：**0** 最大值：**2147483647** 缺省值：**0**

表 1-78 TagV5DTO

参数	参数类型	描述
tag_key	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。
tag_value	String	**参数说明**：标签值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。

请求示例

修改设备，修改设备的名为device，接入类型为安全接入。

```
PUT https://{endpoint}/v5/iot/{project_id}/devices/{device_id}

{
  "device_name" : "device",
  "description" : "watermeter device",
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  },
  "auth_info" : {
    "secure_access" : true
  }
}
```

响应示例

状态码： 200

OK

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "app_name" : "testAPP01",
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
```



```
  "node_id" : "ABC123456789",
  "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "device_name" : "dianadevice",
  "node_type" : "ENDPOINT",
  "description" : "watermeter device",
  "fw_version" : "1.1.0",
  "sw_version" : "1.1.0",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secret" : "3b935a250c50dc2c6d481d048cefdc3c",
    "fingerprint" : "dc0f1016f495157344ac5f1296335cff725ef22f",
    "secure_access" : true,
    "timeout" : 0
  },
  "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
  "product_name" : "Thermometer",
  "status" : "ONLINE",
  "create_time" : "20190303T081011Z",
  "connection_status_update_time" : "2019-03-03T08:10:111Z",
  "active_time" : "2019-03-03T08:10:111Z",
  "tags" : [ {
    "tag_key" : "testTagName",
    "tag_value" : "testTagValue"
  } ],
  "extension_info" : {
    "aaa" : "xxx",
    "bbb" : 0
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改设备，修改设备的名为device，接入类型为安全接入。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
```

```
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateDeviceRequest request = new UpdateDeviceRequest();
        UpdateDevice body = new UpdateDevice();
        AuthInfoWithoutSecret authInfobody = new AuthInfoWithoutSecret();
        authInfobody.withSecureAccess(true);
        body.withAuthInfo(authInfobody);
        body.withExtensionInfo("{\"aaa\":\"xxx\",\"bbb\":0}");
        body.withDescription("watermeter device");
        body.withDeviceName("device");
        request.withBody(body);
        try {
            UpdateDeviceResponse response = client.updateDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改设备，修改设备的名为device，接入类型为安全接入。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()
```



```
    try:
        request = UpdateDeviceRequest()
        authInfobody = AuthInfoWithoutSecret(
            secure_access=True
        )
        request.body = UpdateDevice(
            auth_info=authInfobody,
            extension_info="{\"aaa\":\"xxx\",\"bbb\":0}",
            description="watermeter device",
            device_name="device"
        )
        response = client.update_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改设备，修改设备的名为device，接入类型为安全接入。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateDeviceRequest{}
    secureAccessAuthInfo:= true
    authInfobody := &model.AuthInfoWithoutSecret{
        SecureAccess: &secureAccessAuthInfo,
    }
    var extensionInfoUpdateDevice interface{} = "{\"aaa\":\"xxx\",\"bbb\":0}"
    descriptionUpdateDevice:= "watermeter device"
    deviceNameUpdateDevice:= "device"
```



```
    request.Body = &model.UpdateDevice{
        AuthInfo: authInfobody,
        ExtensionInfo: &extensionInfoUpdateDevice,
        Description: &descriptionUpdateDevice,
        DeviceName: &deviceNameUpdateDevice,
    }
    response, err := client.UpdateDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.5 删除设备

功能介绍

应用服务器可调用此接口在物联网平台上删除指定设备。若设备下连接了非直连设备，则必须把设备下的非直连设备都删除后，才能删除该设备。该接口仅支持删除单个设备，如需批量删除设备，请参见 **创建批量任务**。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/devices/{device_id}

表 1-79 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

请求参数

表 1-80 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除指定设备。

```
DELETE https://{endpoint}/v5/iot/{project_id}/devices/{device_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteDeviceRequest request = new DeleteDeviceRequest();
        try {
            DeleteDeviceResponse response = client.deleteDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```



```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteDeviceRequest()
        response = client.delete_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteDeviceRequest{}
    response, err := client.DeleteDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.6 重置设备密钥

功能介绍

应用服务器可调用此接口重置设备密钥，携带指定密钥时平台将设备密钥重置为指定的密钥，不携带密钥时平台将自动生成一个新的随机密钥返回。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/action

表 1-81 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

表 1-82 Query 参数

参数	是否必选	参数类型	描述
action_id	是	String	**参数说明**：对设备执行的操作。 **取值范围**： • resetSecret: 重置密钥。注意：NB设备密钥由于协议特殊性，只支持十六进制密钥接入。

请求参数

表 1-83 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-84 请求 Body 参数

参数	是否必选	参数类型	描述
secret	否	String	**参数说明**：设备密钥，设置该字段时平台将设备密钥重置为指定值，若不设置则由平台自动生成。 **取值范围**：长度不低于8不超过32，只允许字母、数字、下划线（_）、连接符（-）的组合。 最小长度：**8** 最大长度：**32**
force_disconnect	否	Boolean	**参数说明**：是否强制断开设备的连接，当前仅限长连接。默认值false。 缺省值：**false**
secret_type	否	String	**参数说明**：重置设备密钥的的类型。 **取值范围**： ● PRIMARY：重置主密钥。设备密钥鉴权优先使用的密钥，当设备接入物联网平台时，平台将优先使用主密钥进行校验。 ● SECONDARY：重置辅密钥。设备的备用密钥，当主密钥校验不通过时，会启用辅密钥校验，辅密钥与主密钥有相同的效力；辅密钥对coap协议接入的设备不生效。 缺省值：**PRIMARY**

响应参数

状态码： 200

表 1-85 响应 Body 参数

参数	参数类型	描述
device_id	String	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。 最大长度：**256**

参数	参数类型	描述
secret	String	设备密钥。 最小长度：**8** 最大长度：**32**
secret_type	String	**参数说明**：重置设备密钥的的类型。 **取值范围**： ● PRIMARY：重置主密钥。设备密钥鉴权优先使用的密钥，当设备接入物联网平台时，平台将优先使用主密钥进行校验。 ● SECONDARY：重置辅密钥。设备的备用密钥，当主密钥校验不通过时，会启用辅密钥校验，辅密钥与主密钥有相同的效力；辅密钥对coap协议接入的设备不生效。 缺省值：**PRIMARY**

请求示例

重置指定设备的密钥，新密钥为3b93****dc3c，不强制设备重新建链。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/action?action_id=resetSecret

{
  "secret" : "3b93****dc3c",
  "force_disconnect" : false
}
```

响应示例

状态码： 200

OK

```
{
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "secret" : "3b93****dc3c"
}
```

SDK 代码示例

SDK代码示例如下。

Java

重置指定设备的密钥，新密钥为3b93****dc3c，不强制设备重新建链。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
```



```
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ResetDeviceSecretSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ResetDeviceSecretRequest request = new ResetDeviceSecretRequest();
        request.withActionId("<action_id>");
        ResetDeviceSecret body = new ResetDeviceSecret();
        body.withForceDisconnect(false);
        body.withSecret("3b93****dc3c");
        request.withBody(body);
        try {
            ResetDeviceSecretResponse response = client.resetDeviceSecret(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

重置指定设备的密钥，新密钥为3b93****dc3c，不强制设备重新建链。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
```

```
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ResetDeviceSecretRequest()
        request.action_id = "<action_id>"
        request.body = ResetDeviceSecret(
            force_disconnect=False,
            secret="3b93****dc3c"
        )
        response = client.reset_device_secret(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

重置指定设备的密钥，新密钥为3b93****dc3c，不强制设备重新建链。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
```

```
        WithRegion(region.NewRegion("cn-north-4", endpoint)).
        WithCredential(auth).
        Build())

    request := &model.ResetDeviceSecretRequest{}
    request.ActionId = "<action_id>"
    forceDisconnectResetDeviceSecret:= false
    secretResetDeviceSecret:= "3b93****dc3c"
    request.Body = &model.ResetDeviceSecret{
        ForceDisconnect: &forceDisconnectResetDeviceSecret,
        Secret: &secretResetDeviceSecret,
    }
    response, err := client.ResetDeviceSecret(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.7 冻结设备

功能介绍

应用服务器可调用此接口冻结设备，设备冻结后不能再连接上线，可以通过解冻设备接口解除设备冻结。注意，当前仅支持冻结与平台直连的设备。该接口仅支持冻结单个设备，如需批量冻结设备，请参见 **创建批量任务**。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/freeze

表 1-86 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

请求参数

表 1-87 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

冻结指定设备。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/freeze
```


响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class FreezeDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        FreezeDeviceRequest request = new FreezeDeviceRequest();
        try {
            FreezeDeviceResponse response = client.freezeDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```


Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = FreezeDeviceRequest()
        response = client.freeze_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```



```
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.FreezeDeviceRequest{}
    response, err := client.FreezeDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.8 解冻设备

功能介绍

应用服务器可调用此接口解冻设备，解除冻结后，设备可以连接上线。该接口仅支持解冻单个设备，如需批量解冻设备，请参见 **创建批量任务**。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/unfreeze

表 1-88 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

请求参数

表 1-89 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

解冻指定设备。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/unfreeze
```


响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UnfreezeDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UnfreezeDeviceRequest request = new UnfreezeDeviceRequest();
        try {
            UnfreezeDeviceResponse response = client.unfreezeDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```


Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UnfreezeDeviceRequest()
        response = client.unfreeze_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```



```
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UnfreezeDeviceRequest{}
    response, err := client.UnfreezeDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.9 重置设备指纹

功能介绍

应用服务器可调用此接口重置设备指纹。携带指定设备指纹时将之重置为指定值；不携带时将之置空，后续设备第一次接入时，该设备指纹的值将设置为第一次接入时的证书指纹。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/reset-fingerprint

表 1-90 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

请求参数

表 1-91 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-92 请求 Body 参数

参数	是否必选	参数类型	描述
fingerprint	否	String	**参数说明**：设备指纹。设置该字段时平台将设备指纹重置为指定值；不携带时将之置空，后续设备第一次接入时，该设备指纹的值将设置为第一次接入时的证书指纹。 **取值范围**：长度为40的十六进制字符串或者长度为64的十六进制字符串。
force_disconnect	否	Boolean	**参数说明**：是否强制断开设备的连接，当前仅限长连接。默认值false。 缺省值：**false**
fingerprint_type	否	String	**参数说明**：重置设备证书指纹的的类型。 **取值范围**： ● PRIMARY：重置主指纹。设备证书鉴权优先使用的指纹，当设备接入物联网平台时，平台将优先使用主指纹进行校验。 ● SECONDARY：重置辅指纹。设备的备用指纹，当主指纹校验不通过时，会启用辅指纹校验，辅指纹与主指纹有相同的效力。 缺省值：**PRIMARY**

响应参数

状态码： 200

表 1-93 响应 Body 参数

参数	参数类型	描述
device_id	String	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。 最大长度：**256**
fingerprint	String	设备指纹。

参数	参数类型	描述
fingerprint_type	String	**参数说明**：重置设备证书指纹的的类型。 **取值范围**： ● PRIMARY：重置主指纹。 ● SECONDARY：重置辅指纹。 缺省值：**PRIMARY**

请求示例

重置设备指纹，新指纹为dc0f****f22f。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/reset-fingerprint

{
  "fingerprint" : "dc0f****f22f"
}
```

响应示例

状态码： 200

OK

```
{
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "fingerprint" : "dc0f****f22f"
}
```

SDK 代码示例

SDK代码示例如下。

Java

重置设备指纹，新指纹为dc0f****f22f。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ResetFingerprintSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
```



```
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ResetFingerprintRequest request = new ResetFingerprintRequest();
        ResetFingerprint body = new ResetFingerprint();
        body.withFingerprint("dc0f****f22f");
        request.withBody(body);
        try {
            ResetFingerprintResponse response = client.resetFingerprint(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

重置设备指纹，新指纹为dc0f****f22f。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
```

```
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ResetFingerprintRequest()
        request.body = ResetFingerprint(
            fingerprint="dc0f****f22f"
        )
        response = client.reset_fingerprint(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

重置设备指纹，新指纹为dc0f****f22f。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ResetFingerprintRequest{}
    fingerprintResetFingerprint:= "dc0f****f22f"
    request.Body = &model.ResetFingerprint{
        Fingerprint: &fingerprintResetFingerprint,
    }
    response, err := client.ResetFingerprint(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.10 灵活搜索设备列表

功能介绍

接口说明

应用服务器使用SQL语句调用该接口，灵活的搜索所需要的设备资源列表

限制

- 仅**标准版实例、企业版实例**支持该接口调用，基础版不支持。
- 单账号调用该接口的 TPS 限制最大为1/S(每秒1次请求数)

类SQL语法使用说明

类SQL语句有select、from、where(可选)、order by(可选)、limit子句(可选)组成，长度限制为400个字符。子句里的内容大小写敏感，SQL语句的关键字大小写不敏感。

示例：

```
select * from device where device_id = 'as********' limit 0,5
```

SELECT子句

```
select [field]/[count(*)/count(1)] from device
```

其中field为需要获取的字段，请参考响应参数字段名称，也可填*，获取所有字段。

如果需要统计搜索的设备个数，请填count(*)或者count(1).

FROM子句

```
from device
```

from后为要查询的资源名，当前支持"device"

WHERE子句(可选)

```
WHERE [condition1] AND [condition2]
```

最多支持5个condition，不支持嵌套；支持的检索字段请参见下面的**搜索条件字段说明**和**支持的运算符**章节

连接词支持AND、OR，优先级参考标准SQL语法，默认AND优先级高于OR。

LIMIT子句(可选)

```
limit [offset,] rows
```

offset标识搜索的偏移量，rows标识返回搜索结果的最大行数，例如：

- limit n ;示例(select * from device limit 10)
  最大返回n条结果数据
- limit m,n; 示例(select * from device limit 20,10) 搜索偏移量为m，最大返回n条结果数据

限制

offset 最大 500， rows最大50，如果不填写limit子句，默认为limit 10

ORDER BY子句(可选)

用于实现自定义排序，当前支持自定义排序的字段为："marker"。

```
order by marker [asc]/[desc]
```

子句不填写时默认逻辑为随机排序

搜索条件字段说明

字段名	类型	说明	取值范围
app_id	string	资源空间ID	长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
device_id	string	设备ID	长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。
gateway_id	string	网关ID	长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。
product_id	string	设备关联的产品ID	长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

字段名	类型	说明	取值范围
device_name	string	设备名称	长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合符。
node_id	string	设备标识码	长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合
status	string	设备的状态	ONLINE(在线)、OFFLINE(离线)、ABNORMAL(异常)、INACTIVE(未激活)、FROZEN(冻结)
node_type	string	设备节点类型	GATEWAY(直连设备或网关)、ENDPOINT(非直连设备)
tag_key	string	标签键	长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。
tag_value	string	标签值	长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。
sw_version	string	软件版本	长度不超过64，只允许字母、数字、下划线（_）、连接符（-）、英文点(.)的组合。
fw_version	string	固件版本	长度不超过64，只允许字母、数字、下划线（_）、连接符（-）、英文点(.)的组合。
group_id	string	群组Id	长度不超过36，十六进制字符串和连接符（-）的组合

字段名	类型	说明	取值范围
create_time	string	设备注册时间	格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'，如：2015-06-06T12:10:10.000Z
marker	string	结果记录ID	长度为24的十六进制字符串，如ffffffffffffffffffffffff

支持的运算符

运算符	支持的字段
=	所有
!=	所有
>	create_time、marker
<	create_time、marker
like	device_name、node_id、tag_key、tag_value
in	除tag_key、tag_valuel以外字段
not in	除tag_key、tag_valuel以外字段

SQL 限制

- like: 只支持前缀匹配，不支持后缀匹配或者通配符匹配。前缀匹配不得少于4个字符，且不能包含任何特殊字符(只允许中文、字母、数字、下划线（_）、连接符（-）). 前缀后必须跟上"%"结尾。
- 不支持除了count(*)/count(1)以外的其他任何函数。
- 不支持其他SQL用法，如嵌套SQL、union、join、别名(Alias)等用法
- SQL长度限制为400个字符，单个请求条件最大支持5个。
- 不支持"null"和空字符串等条件值匹配

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/search/query-devices

表 1-94 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-95 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-96 请求 Body 参数

参数	是否必选	参数类型	描述
sql	是	String	搜索sql语句，具体使用方法见类SQL语法使用说明章节 最小长度：**1** 最大长度：**400**

响应参数

状态码： 200

表 1-97 响应 Body 参数

参数	参数类型	描述
devices	Array of **SearchDevice** objects	搜索设备结果列表。
count	Long	满足查询条件的记录总数(只有条件为select count(*)/count(1)时单独返回)。

表 1-98 SearchDevice

参数	参数类型	描述
app_id	String	资源空间ID。 最大长度：**36**
app_name	String	资源空间名称。
device_id	String	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。 最大长度：**256**
node_id	String	设备标识码，通常使用IMEI、MAC地址或Serial No作为node_id。 最大长度：**64**
gateway_id	String	网关ID，用于标识设备所属的父设备，即父设备的设备ID。当设备是直连设备时，gateway_id与设备的device_id一致。当设备是非直连设备时，gateway_id为设备所关联的父设备的device_id。 最大长度：**256**
device_name	String	设备名称。 最大长度：**256**
node_type	String	设备节点类型。 ● ENDPOINT：非直连设备。 ● GATEWAY：直连设备或网关。 ● UNKNOWN：未知。
fw_version	String	设备的固件版本。 最大长度：**256**
sw_version	String	设备的软件版本。 最大长度：**256**

参数	参数类型	描述
device_sdk_version	String	设备的sdk信息。 最大长度：**256**
product_id	String	设备关联的产品ID，用于唯一标识一个产品模型。
product_name	String	设备关联的产品名称。
groups	Object	设备组列表。
status	String	设备的状态。 ● ONLINE：设备在线。 ● OFFLINE：设备离线。 ● ABNORMAL：设备异常。 ● INACTIVE：设备未激活。 ● FROZEN：设备冻结。
tags	Object	设备的标签列表。
marker	String	搜索结果记录Id。

请求示例

通过sql查询设备，查询所有设备。

```
POST https://{endpoint}/v5/iot/{project_id}/search/query-devices

{
  "sql" : "select * from device"
}
```

响应示例

状态码： 200

OK

```
{
  "devices" : [ {
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "marker" : "5c8f3d2d3df1f10d803adbda",
    "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "node_id" : "ABC123456789",
    "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "device_name" : "dianadevice",
    "node_type" : "ENDPOINT",
    "fw_version" : "1.1.0",
    "sw_version" : "1.1.0",
    "device_sdk_version" : "1.1.0",
    "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
    "status" : "INACTIVE",
    "tags" : [ {
      "tag_key" : "testTagName",
      "tag_value" : "testTagValue"
    } ]
```



```
 }]
}
```

SDK 代码示例

SDK代码示例如下。

Java

通过sql查询设备，查询所有设备。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class SearchDevicesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        SearchDevicesRequest request = new SearchDevicesRequest();
        SearchSql body = new SearchSql();
        body.withSql("select * from device");
        request.withBody(body);
        try {
            SearchDevicesResponse response = client.searchDevices(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
```



```
        }
    }
}
```

Python

通过sql查询设备，查询所有设备。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = SearchDevicesRequest()
        request.body = SearchSql(
            sql="select * from device"
        )
        response = client.search_devices(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

通过sql查询设备，查询所有设备。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
```

```
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.SearchDevicesRequest{}
    request.Body = &model.SearchSql{
        Sql: "select * from device",
    }
    response, err := client.SearchDevices(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.2.11 查询指定设备加入的设备组列表

功能介绍

应用服务器可调用此接口查询物联网平台中的某个设备加入的设备组信息列表。仅标准版实例、企业版实例支持该接口调用，基础版不支持。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/list-device-group

表 1-99 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

请求参数

表 1-100 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-101 响应 Body 参数

参数	参数类型	描述
device_groups	Array of **ListDeviceGroupSummary** objects	设备组信息列表。

表 1-102 ListDeviceGroupSummary

参数	参数类型	描述
group_id	String	设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。
name	String	设备组名称，单个资源空间下不可重复。
description	String	设备组描述。
super_group_id	String	父设备组ID，该设备组的父设备组ID。
group_type	String	**参数说明**：设备组类型，默认为静态设备组；当设备组类型为动态设备组时，需要填写动态设备组规则

请求示例

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/list-device-group
```

响应示例

状态码： 200

OK

```
{
  "device_groups" : [ {
    "group_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
    "name" : "GroupA",
    "description" : "群组A",
    "super_group_id" : "04ed32dc1b0025b52fe3c01a27c2b0a8",
    "group_type" : "STATIC"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListDeviceGroupsByDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListDeviceGroupsByDeviceRequest request = new ListDeviceGroupsByDeviceRequest();
        try {
            ListDeviceGroupsByDeviceResponse response = client.listDeviceGroupsByDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```



```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListDeviceGroupsByDeviceRequest()
        response = client.list_device_groups_by_device(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListDeviceGroupsByDeviceRequest{}
    response, err := client.ListDeviceGroupsByDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.3 设备消息

1.4.3.1 下发设备消息

功能介绍

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。平台返回应用响应结果不一定是设备接收结果，建议用户应用通过订阅**设备消息状态变更通知**，订阅后平台会将设备接收结果推送给订阅的应用。 注意：

- 此接口适用于MQTT设备消息下发，暂不支持其他协议接入的设备消息下发。
- 此接口仅支持单个设备消息下发，如需多个设备消息下发，请参见 **创建批量任务**。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/messages

表 1-103 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**参数说明**：下发消息的设备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。

请求参数

表 1-104 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-105 请求 Body 参数

参数	是否必选	参数类型	描述
message_id	否	String	**参数说明**：消息id，由用户生成（推荐使用UUID），同一个设备下唯一， 如果用户填写的id在设备下不唯一， 则接口返回错误。 **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。 最大长度：**128**
name	否	String	**参数说明**：消息名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最大长度：**128**
message	是	Object	**参数说明**：消息内容，支持string和json格式。
properties	否	**PropertiesDTO** object	**参数说明**：消息下行到设备的属性参数。
encoding	否	String	**参数说明**：消息内容编码格式。默认值none。 **取值范围**： • none • base64 缺省值：**none**
payload_format	否	String	**参数说明**：有效负载格式，在消息内容编码格式为none时有效。默认值standard（平台封装的标准格式）。 **取值范围**： • standard • raw：直接将消息内容作为有效负载下发。 缺省值：**standard**

参数	是否必选	参数类型	描述
topic	否	String	**参数说明**：消息下行到设备的自定义topic后缀, 可选， 仅适用于MQTT协议接入的设备。 用户只能填写在租户产品界面配置的topic, 否则会校验不通过。 平台给消息topic添加的前缀为$oc/devices/{device_id}/user/， 用户可以在前缀的基础上增加自定义部分， 如增加messageDown，则平台拼接前缀后完整的topic为 $oc/devices/{device_id}/user/messageDown，其中device_id以实际设备的网关id替代。 如果用户指定该topic，消息会通过该topic下行到设备，如果用户不指定， 则消息通过系统默认的topic下行到设备,系统默认的topic格式为： $oc/devices/{device_id}/sys/messages/down。此字段与topic_full_name字段只可填写一个。 最大长度：**128**
topic_full_name	否	String	**参数说明**：消息下行到设备的完整topic名称, 可选。用户需要下发用户自定义的topic给设备时，可以使用该参数指定完整的topic名称，物联网平台不校验该topic是否在平台定义，直接透传给设备。设备需要提前订阅该topic。此字段与topic字段只可填写一个。 最大长度：**128**
ttl	否	Integer	**参数说明**：下发消息在平台缓存的老化时间，时间单位是分钟，默认值1440；ttl参数数值必须是5的倍数，即以5分钟为粒度；指定为0时表示不缓存消息，默认最大缓存时间为1440分钟，缓存时间超过1440分钟需申请，否则下发失败。 最小值：**0** 最大值：**10080** 缺省值：**1440**

表 1-106 PropertiesDTO

参数	是否必选	参数类型	描述
correlation_data	否	String	**参数说明**：MQTT 5.0版本请求和响应模式中的相关数据，可选。用户可以通过该参数配置MQTT协议请求和响应模式中的相关数据。 **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。
response_topic	否	String	**参数说明**：MQTT 5.0版本请求和响应模式中的响应主题，可选。用户可以通过该参数配置MQTT协议请求和响应模式中的响应主题。 **取值范围**：长度不超过128, 只允许字母、数字、以及_-?=$#+/等字符的组合。 最大长度：**128**
user_properties	否	Array of **UserPropDTO** objects	**参数说明**：用户自定义属性，可选。用户可以通过该参数配置用户自定义属性。可以配置的最大自定义属性数量为20。

表 1-107 UserPropDTO

参数	是否必选	参数类型	描述
prop_key	否	String	**参数说明**：用户自定义属性键。 **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。
prop_value	否	String	**参数说明**：用户自定义属性值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

响应参数

状态码： 201

表 1-108 响应 Body 参数

参数	参数类型	描述
message_id	String	消息id，由用户生成（推荐使用UUID），同一个设备下唯一， 如果用户不填写，则由物联网平台生成。
result	**MessageResult** object	消息下发结果。Json格式。

表 1-109 MessageResult

参数	参数类型	描述
status	String	消息状态, PENDING，DELIVERED，FAILED和TIMEOUT。如果设备不在线，则平台缓存消息，并且返回PENDING，等设备数据上报之后再下发；如果设备在线，则消息直接进行下发，下发成功后接口返回DELIVERED，失败返回FAILED；如果消息在平台默认时间内（1天）还没有下发给设备，则平台会将消息设置为超时，状态为TIMEOUT。另外应用可以订阅消息的执行结果，平台会将消息结果推送给订阅的应用。
created_time	String	消息的创建时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
finished_time	String	消息结束时间, "yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串，包含消息转换到DELIVERED，FAILED和TIMEOUT状态的时间。

请求示例

- 创建消息，通过平台默认topic下发。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/messages

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

- 创建消息，通过平台自定义topic下发。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/messages

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

- 创建消息，通过自定义topic下发。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/messages
```



```
{
  "message_id" : "99b32da9-cd17-4cdf-a286-f6e849cbc364",
  "name" : "messageName",
  "message" : "HelloWorld",
  "topic_full_name" : "/test/customTopic/testTopic"
}
```

响应示例

状态码： 201

Created

```
{
  "message_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "result" : {
    "status" : "PENDING",
    "created_time" : "20151212T121212Z",
    "finished_time" : "20151212T121213Z"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建消息，通过平台默认topic下发。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
```

```
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        CreateMessageRequest request = new CreateMessageRequest();
        DeviceMessageRequest body = new DeviceMessageRequest();
        body.withMessage("HelloWorld");
        body.withName("messageName");
        body.withMessageId("99b32da9-cd17-4cdf-a286-f6e849cbc364");
        request.withBody(body);
        try {
            CreateMessageResponse response = client.createMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建消息，通过平台自定义topic下发。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateMessageRequest request = new CreateMessageRequest();
        DeviceMessageRequest body = new DeviceMessageRequest();
        body.withTopic("testTopic");
```

```
        body.withMessage("HelloWorld");
        body.withName("messageName");
        body.withMessageId("99b32da9-cd17-4cdf-a286-f6e849cbc364");
        request.withBody(body);
        try {
            CreateMessageResponse response = client.createMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建消息，通过自定义topic下发。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateMessageRequest request = new CreateMessageRequest();
        DeviceMessageRequest body = new DeviceMessageRequest();
        body.withTopicFullName("/test/customTopic/testTopic");
        body.withMessage("HelloWorld");
        body.withName("messageName");
        body.withMessageId("99b32da9-cd17-4cdf-a286-f6e849cbc364");
        request.withBody(body);
        try {
```

```
            CreateMessageResponse response = client.createMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建消息，通过平台默认topic下发。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateMessageRequest()
        request.body = DeviceMessageRequest(
            message="HelloWorld",
            name="messageName",
            message_id="99b32da9-cd17-4cdf-a286-f6e849cbc364"
        )
        response = client.create_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建消息，通过平台自定义topic下发。

```
# coding: utf-8
```

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateMessageRequest()
        request.body = DeviceMessageRequest(
            topic="testTopic",
            message="HelloWorld",
            name="messageName",
            message_id="99b32da9-cd17-4cdf-a286-f6e849cbc364"
        )
        response = client.create_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建消息，通过自定义topic下发。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```



```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateMessageRequest()
        request.body = DeviceMessageRequest(
            topic_full_name="/test/customTopic/testTopic",
            message="HelloWorld",
            name="messageName",
            message_id="99b32da9-cd17-4cdf-a286-f6e849cbc364"
        )
        response = client.create_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建消息，通过平台默认topic下发。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateMessageRequest{}
    var messageDeviceMessageRequest interface{} = "HelloWorld"
    nameDeviceMessageRequest:= "messageName"
    messageIdDeviceMessageRequest:= "99b32da9-cd17-4cdf-a286-f6e849cbc364"
    request.Body = &model.DeviceMessageRequest{
```

```
        Message: &messageDeviceMessageRequest,
        Name: &nameDeviceMessageRequest,
        MessageId: &messageIdDeviceMessageRequest,
    }
    response, err := client.CreateMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建消息，通过平台自定义topic下发。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateMessageRequest{}
    topicDeviceMessageRequest:= "testTopic"
    var messageDeviceMessageRequest interface{} = "HelloWorld"
    nameDeviceMessageRequest:= "messageName"
    messageIdDeviceMessageRequest:= "99b32da9-cd17-4cdf-a286-f6e849cbc364"
    request.Body = &model.DeviceMessageRequest{
        Topic: &topicDeviceMessageRequest,
        Message: &messageDeviceMessageRequest,
        Name: &nameDeviceMessageRequest,
        MessageId: &messageIdDeviceMessageRequest,
    }
    response, err := client.CreateMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


- 创建消息，通过自定义topic下发。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateMessageRequest{}
    topicFullNameDeviceMessageRequest:= "/test/customTopic/testTopic"
    var messageDeviceMessageRequest interface{} = "HelloWorld"
    nameDeviceMessageRequest:= "messageName"
    messageIdDeviceMessageRequest:= "99b32da9-cd17-4cdf-a286-f6e849cbc364"
    request.Body = &model.DeviceMessageRequest{
        TopicFullName: &topicFullNameDeviceMessageRequest,
        Message: &messageDeviceMessageRequest,
        Name: &nameDeviceMessageRequest,
        MessageId: &messageIdDeviceMessageRequest,
    }
    response, err := client.CreateMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.3.2 查询设备消息

功能介绍

应用服务器可调用此接口查询平台下发给设备的消息，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。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/devices/{device_id}/messages

表 1-110 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**参数说明**：下发消息的设备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。

请求参数

表 1-111 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-112 响应 Body 参数

参数	参数类型	描述
device_id	String	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。
messages	Array of **DeviceMessage** objects	设备消息列表。

表 1-113 DeviceMessage

参数	参数类型	描述
message_id	String	设备消息ID，用于唯一标识一条消息，在下发设备消息时由物联网平台分配获得。
name	String	消息名称,在下发消息时由用户指定。
message	Object	消息内容。
encoding	String	消息内容编码格式，取值范围none|base64,默认值none, base64格式仅支持透传。

参数	参数类型	描述
payload_format	String	有效负载格式，在消息内容编码格式为none时有效，取值范围standard|raw，默认值standard（平台封装的标准格式），取值为raw时直接将消息内容作为有效负载下发。
topic	String	消息topic
properties	**PropertiesDTO** object	消息下行到设备的属性参数。
status	String	消息状态，包含PENDING，DELIVERED，FAILED和TIMEOUT，PENDING指设备不在线，消息被缓存起来，等设备上线之后下发；DELIVERED指消息发送成功；FAILED消息发送失败；TIMEOUT指消息在平台默认时间内（1天）还没有下发送给设备，则平台会将消息设置为超时，状态为TIMEOUT。
error_info	**ErrorInfoDTO** object	消息下发失败信息。
created_time	String	消息的创建时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
finished_time	String	消息结束时间, "yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串，包含消息转换到DELIVERED和TIMEOUT两个状态的时间。

表 1-114 PropertiesDTO

参数	参数类型	描述
correlation_data	String	**参数说明**：MQTT 5.0版本请求和响应模式中的相关数据，可选。用户可以通过该参数配置MQTT协议请求和响应模式中的相关数据。 **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。
response_topic	String	**参数说明**：MQTT 5.0版本请求和响应模式中的响应主题，可选。用户可以通过该参数配置MQTT协议请求和响应模式中的响应主题。 **取值范围**：长度不超过128, 只允许字母、数字、以及_-?=$#+/等字符的组合。 最大长度：**128**
user_properties	Array of **UserPropDTO** objects	**参数说明**：用户自定义属性，可选。用户可以通过该参数配置用户自定义属性。可以配置的最大自定义属性数量为20。

表 1-115 UserPropDTO

参数	参数类型	描述
prop_key	String	**参数说明**：用户自定义属性键。 **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。
prop_value	String	**参数说明**：用户自定义属性值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

表 1-116 ErrorInfoDTO

参数	参数类型	描述
error_code	String	**参数说明**：异常信息错误码，包含IOTDA.014016和IOTDA.014112。IOTDA.014016表示设备不在线；IOTDA.014112表示设备没有订阅topic。
error_msg	String	**参数说明**：异常信息说明，包含设备不在线和设备没有订阅topic说明。

请求示例

列表查询所有消息。

```
GET https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/messages
```

响应示例

状态码： 200

OK

```
{
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "messages" : [ {
    "message_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
    "name" : "message_name",
    "message" : "string",
    "topic" : "string",
    "status" : "PENDING",
    "created_time" : "20151212T121212Z",
    "finished_time" : "20151212T121212Z"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;
```



```
import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListDeviceMessagesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListDeviceMessagesRequest request = new ListDeviceMessagesRequest();
        try {
            ListDeviceMessagesResponse response = client.listDeviceMessages(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
```

```
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListDeviceMessagesRequest()
        response = client.list_device_messages(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListDeviceMessagesRequest{}
    response, err := client.ListDeviceMessages(request)
```

```
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.3.3 查询指定消息 id 的消息

功能介绍

应用服务器可调用此接口查询平台下发给设备的指定消息id的消息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/devices/{device_id}/messages/{message_id}

表 1-117 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

参数	是否必选	参数类型	描述
device_id	是	String	**参数说明**：下发消息的设备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。
message_id	是	String	**参数说明**：下发消息的消息ID，用于唯一标识一个消息，在消息下发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。

请求参数

表 1-118 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-119 响应 Body 参数

参数	参数类型	描述
message_id	String	设备消息ID，用于唯一标识一条消息，在下发设备消息时由物联网平台分配获得。

参数	参数类型	描述
name	String	消息名称,在下发消息时由用户指定。
message	Object	消息内容。
encoding	String	消息内容编码格式，取值范围none|base64,默认值none, base64格式仅支持透传。
payload_format	String	有效负载格式，在消息内容编码格式为none时有效，取值范围standard|raw，默认值standard（平台封装的标准格式），取值为raw时直接将消息内容作为有效负载下发。
topic	String	消息topic
properties	**PropertiesDTO** object	消息下行到设备的属性参数。
status	String	消息状态，包含PENDING，DELIVERED，FAILED和TIMEOUT，PENDING指设备不在线，消息被缓存起来，等设备上线之后下发；DELIVERED指消息发送成功；FAILED消息发送失败；TIMEOUT指消息在平台默认时间内（1天）还没有下发送给设备，则平台会将消息设置为超时，状态为TIMEOUT。
error_info	**ErrorInfoDTO** object	消息下发失败信息。
created_time	String	消息的创建时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
finished_time	String	消息结束时间, "yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串，包含消息转换到DELIVERED和TIMEOUT两个状态的时间。

表 1-120 PropertiesDTO

参数	参数类型	描述
correlation_data	String	**参数说明**：MQTT 5.0版本请求和响应模式中的相关数据，可选。用户可以通过该参数配置MQTT协议请求和响应模式中的相关数据。 **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。
response_topic	String	**参数说明**：MQTT 5.0版本请求和响应模式中的响应主题，可选。用户可以通过该参数配置MQTT协议请求和响应模式中的响应主题。 **取值范围**：长度不超过128, 只允许字母、数字、以及_-?=$#+/等字符的组合。 最大长度：**128**

参数	参数类型	描述
user_properties	Array of UserPropDTO objects	**参数说明**：用户自定义属性，可选。用户可以通过该参数配置用户自定义属性。可以配置的最大自定义属性数量为20。

表 1-121 UserPropDTO

参数	参数类型	描述
prop_key	String	**参数说明**：用户自定义属性键。 **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。
prop_value	String	**参数说明**：用户自定义属性值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。

表 1-122 ErrorInfoDTO

参数	参数类型	描述
error_code	String	**参数说明**：异常信息错误码，包含IOTDA.014016和IOTDA.014112。IOTDA.014016表示设备不在线；IOTDA.014112表示设备没有订阅topic。
error_msg	String	**参数说明**：异常信息说明，包含设备不在线和设备没有订阅topic说明。

请求示例

查询指定的消息详情。

```
GET https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/messages/{message_id}
```

响应示例

状态码： 200

OK

```
{
  "message_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "name" : "message_name",
  "message" : "string",
  "topic" : "string",
  "status" : "PENDING",
  "created_time" : "20151212T121212Z",
  "finished_time" : "20151212T121212Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDeviceMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDeviceMessageRequest request = new ShowDeviceMessageRequest();
        try {
            ShowDeviceMessageResponse response = client.showDeviceMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```



```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDeviceMessageRequest()
        response = client.show_device_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```

```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDeviceMessageRequest{}
    response, err := client.ShowDeviceMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.4 设备命令

1.4.4.1 设备同步命令

1.4.4.1.1 下发设备命令

功能介绍

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，平台超时时间是20秒。如果命令下发需要超过20秒，建议采用**消息下发**。 注意：

- 此接口适用于MQTT设备同步命令下发，暂不支持NB-IoT设备命令下发。
- 此接口仅支持单个设备同步命令下发，如需多个设备同步命令下发，请参见 **创建批量任务**。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/commands

表 1-123 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**参数说明**：下发消息的设备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。

请求参数

表 1-124 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-125 请求 Body 参数

参数	是否必选	参数类型	描述
service_id	否	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。 **取值范围**：长度不超过64的字符串。 最大长度：**64**
command_name	否	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。 **取值范围**：长度不超过128的字符串。 最大长度：**128**
paras	是	Object	**参数说明**：设备执行的命令，Json格式，里面是一个个键值对，如果serviceId不为空，每个键都是profile中命令的参数名（paraName）;如果serviceId为空则由用户自定义命令格式。设备命令示例：{"value":"1"}，具体格式需要应用和设备约定。此参数仅支持Json格式，暂不支持字符串。 最大长度：**261952**

响应参数

状态码： 200

表 1-126 响应 Body 参数

参数	参数类型	描述
command_id	String	设备命令ID，用于唯一标识一条命令，在下发设备命令时由物联网平台分配获得。
response	Object	设备上报的命令执行结果。Json格式，具体格式需要应用和设备约定。
error_code	String	命令下发异常错误码。
error_msg	String	命令下发异常错误信息。

请求示例

创建命令，命令名为ON_OFF，命令为ON。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/commands
```



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

响应示例

状态码： 200

OK

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

SDK 代码示例

SDK代码示例如下。

Java

创建命令，命令名为ON_OFF，命令为ON。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateCommandSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);
```



```
        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateCommandRequest request = new CreateCommandRequest();
        DeviceCommandRequest body = new DeviceCommandRequest();
        body.withParas("{\"value\":\"ON\"}");
        body.withCommandName("ON_OFF");
        body.withServiceId("reboot");
        request.withBody(body);
        try {
            CreateCommandResponse response = client.createCommand(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建命令，命令名为ON_OFF，命令为ON。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateCommandRequest()
        request.body = DeviceCommandRequest(
            paras="{\"value\":\"ON\"}",
            command_name="ON_OFF",
            service_id="reboot"
```



```
        )
        response = client.create_command(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建命令，命令名为ON_OFF，命令为ON。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateCommandRequest{}
    var parasDeviceCommandRequest interface{} = "{\"value\":\"ON\"}"
    commandNameDeviceCommandRequest:= "ON_OFF"
    serviceIdDeviceCommandRequest:= "reboot"
    request.Body = &model.DeviceCommandRequest{
        Paras: &parasDeviceCommandRequest,
        CommandName: &commandNameDeviceCommandRequest,
        ServiceId: &serviceIdDeviceCommandRequest,
    }
    response, err := client.CreateCommand(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.4.2 设备异步命令

1.4.4.2.1 下发异步设备命令

功能介绍

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异步命令，以实现对设备的控制。平台负责将命令发送给设备，并将设备执行命令结果异步通知应用服务器。 命令执行结果支持灵活的数据流转，应用服务器通过调用物联网平台的创建规则触发条件（Resource:device.command.status，Event:update）、创建规则动作并激活规则后，当命令状态变更时，物联网平台会根据规则将结果发送到规则指定的服务器，如用户自定义的HTTP服务器，AMQP服务器，以及华为云的其他存储服务器等, 详情参考**设备命令状态变更通知**。 注意：

- 此接口适用于NB设备异步命令下发，暂不支持其他协议类型设备命令下发。
- 此接口仅支持单个设备异步命令下发，如需多个设备异步命令下发，请参见 **创建批量任务**。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/devices/{device_id}/async-commands

表 1-127 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**参数说明**：下发命令的设备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。

请求参数

表 1-128 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-129 请求 Body 参数

参数	是否必选	参数类型	描述
service_id	否	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。如设备需要编解码插件来解析命令，此参数为必填项。 **取值范围**：长度不超过64的字符串。 最大长度：**64**

参数	是否必选	参数类型	描述
command_name	否	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。如设备需要编解码插件来解析命令，此参数为必填项。 **取值范围**：长度不超过128的字符串。 最大长度：**128**
paras	是	Object	**参数说明**：设备执行的命令，Json格式，里面是一个个健值对，如果service_id不为空，每个健都是profile中命令的参数名（paraName）;如果service_id为空则由用户自定义命令格式。设备命令示例：{"value":"1"}，具体格式需要应用和设备约定，请求对象最大长度为256KB 最大长度：**261909**
expire_time	否	Integer	**参数说明**：物联网平台缓存命令的时长， 单位秒, 平台最多缓存20条消息（即最多缓存20条PENDING状态的命令） 该参数在send_strategy字段为delay时有效，默认缓存24小时，最大缓存2天。 最小值：**0** 最大值：**172800**
send_strategy	是	String	**参数说明**：下发策略，默认缓存下发。 **取值范围**： • immediately:表示立即下发，此时expire_time无效。 • delay:表示缓存下发，等数据上报或者设备上线之后下发。expire_time为0或空时，命令会默认缓存24小时。

响应参数

状态码： 200

表 1-130 响应 Body 参数

参数	参数类型	描述
device_id	String	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。

参数	参数类型	描述
command_id	String	设备命令ID，用于唯一标识一条命令，在下发设备命令时由物联网平台分配获得。
service_id	String	设备命令所属的设备服务ID，在设备关联的产品模型中定义。
command_name	String	设备命令名称，在设备关联的产品模型中定义。
paras	Object	设备执行的命令，Json格式，里面是一个个健值对，如果service_id不为空，每个健都是profile中命令的参数名（paraName）;如果service_id为空则由用户自定义命令格式。设备命令示例：{"value":"1"}，具体格式需要应用和设备约定。
expire_time	Integer	物联网平台缓存命令的时长， 单位秒。
status	String	设备命令状态,如果命令被缓存，返回PENDING,如果命令下发给设备，返回SENT。
created_time	String	命令的创建时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
send_strategy	String	下发策略， immediately表示立即下发，delay表示缓存起来，等数据上报或者设备上线之后下发。

请求示例

创建异步命令，命令名为ON_OFF，命令为ON。

```
POST https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/async-commands

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

响应示例

状态码： 200

OK

```
{
  "device_id" : "c1224afb-e9f0-4916-8220-b6bab568e888",
  "command_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "service_id" : "Switch",
  "command_name" : "ON_OFF",
  "send_strategy" : "immediately",
  "paras" : {
    "value" : "ON"
```



```
 },
 "expire_time" : 0,
 "status" : "SENT",
 "created_time" : "20151212T121212Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建异步命令，命令名为ON_OFF，命令为ON。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateAsyncCommandSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateAsyncCommandRequest request = new CreateAsyncCommandRequest();
        AsyncDeviceCommandRequest body = new AsyncDeviceCommandRequest();
        body.withSendStrategy("immediately");
        body.withExpireTime(0);
        body.withParas("{\"value\":\"ON\"}");
        body.withCommandName("ON_OFF");
        body.withServiceId("reboot");
        request.withBody(body);
        try {
            CreateAsyncCommandResponse response = client.createAsyncCommand(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
```

```
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建异步命令，命令名为ON_OFF，命令为ON。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateAsyncCommandRequest()
        request.body = AsyncDeviceCommandRequest(
            send_strategy="immediately",
            expire_time=0,
            paras="{\"value\":\"ON\"}",
            command_name="ON_OFF",
            service_id="reboot"
        )
        response = client.create_async_command(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建异步命令，命令名为ON_OFF，命令为ON。

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateAsyncCommandRequest{}
    expireTimeAsyncDeviceCommandRequest:= int32(0)
    var parasAsyncDeviceCommandRequest interface{} = "{\"value\":\"ON\"}"
    commandNameAsyncDeviceCommandRequest:= "ON_OFF"
    serviceIdAsyncDeviceCommandRequest:= "reboot"
    request.Body = &model.AsyncDeviceCommandRequest{
        SendStrategy: "immediately",
        ExpireTime: &expireTimeAsyncDeviceCommandRequest,
        Paras: &parasAsyncDeviceCommandRequest,
        CommandName: &commandNameAsyncDeviceCommandRequest,
        ServiceId: &serviceIdAsyncDeviceCommandRequest,
    }
    response, err := client.CreateAsyncCommand(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK

状态码	描述
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.4.2.2 查询指定 id 的命令

功能介绍

物联网平台可查询指定id的命令。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/devices/{device_id}/async-commands/{command_id}

表 1-131 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**参数说明**：下发命令的设备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。
command_id	是	String	**参数说明**：下发命令的命令id，用于唯一标识一个消息，在下发命令时由物联网平台分配获得。 **取值范围**：长度不超过100，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-132 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-133 响应 Body 参数

参数	参数类型	描述
device_id	String	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。
command_id	String	设备命令ID，用于唯一标识一条命令，在下发设备命令时由物联网平台分配获得。
service_id	String	设备命令所属的设备服务ID，在设备关联的产品模型中定义。
command_name	String	设备命令名称，在设备关联的产品模型中定义。
paras	Object	设备执行的命令，Json格式，里面是一个个健值对，如果service_id不为空，每个健都是profile中命令的参数名（paraName）;如果service_id为空则由用户自定义命令格式。设备命令示例：{"value":"1"}，具体格式需要应用和设备约定。
expire_time	Integer	物联网平台缓存命令的时长， 单位秒。

参数	参数类型	描述
status	String	下发命令的状态。 ·PENDING表示未下发,在物联网平台缓存着 ·EXPIRED表示命令已经过期，即缓存的时间超过设定的expire_time ·SENT表示命令正在下发 ·DELIVERED表示命令已送达设备 · SUCCESSFUL表示命令已经成功执行 ·FAILED表示命令执行失败 ·TIMEOUT表示命令下发之后，没有收到设备确认或者响应结果一定时间后超时
result	Object	设备命令执行的详细结果，由设备返回，Json格式。
created_time	String	命令的创建时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
sent_time	String	物联网平台发送命令的时间，如果命令是立即下发， 则该时间与命令创建时间一致， 如果是缓存命令， 则是命令实际下发的时间。 "yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
delivered_time	String	物联网平台将命令送达到设备的时间， "yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
send_strategy	String	下发策略， immediately表示立即下发，delay表示缓存起来，等数据上报或者设备上线之后下发。
response_time	String	设备响应命令的时间， "yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。

请求示例

查询指定id的命令。

```
GET https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/async-commands/{command_id}
```

响应示例

状态码： 200

OK

```
{
  "device_id" : "c1224afb-e9f0-4916-8220-b6bab568e888",
  "command_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "service_id" : "Switch",
  "command_name" : "ON_OFF",
  "paras" : {
    "value" : "ON"
  },
  "expire_time" : 0,
  "send_strategy" : "immediately",
  "created_time" : "20151212T121212Z",
  "status" : "DELIVERED",
  "result" : {
    "code" : 200
  },
```



```
  "sent_time" : "20151212T121212Z",
  "delivered_time" : "20151212T121212Z",
  "response_time" : "20151212T131312Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowAsyncDeviceCommandSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowAsyncDeviceCommandRequest request = new ShowAsyncDeviceCommandRequest();
        try {
            ShowAsyncDeviceCommandResponse response = client.showAsyncDeviceCommand(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```


Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowAsyncDeviceCommandRequest()
        response = client.show_async_device_command(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```

```
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowAsyncDeviceCommandRequest{}
    response, err := client.ShowAsyncDeviceCommand(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.5 设备属性

1.4.5.1 修改设备属性

功能介绍

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发属性。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。 注意：此接口适用于MQTT设备，暂不支持NB-IoT设备。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/devices/{device_id}/properties

表 1-134 路径参数

参数	是否必选	参数类型	描述
device_id	是	String	**参数说明**：下发属性的设备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。
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-135 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-136 请求 Body 参数

参数	是否必选	参数类型	描述
services	否	Object	**参数说明**：设备执行的属性，Json格式，里面是一个个键值对，如果serviceId不为空，每个键都是profile中属性的参数名（paraName）;如果serviceId为空则由用户自定义属性格式。设属性示例：[{"service_id": "Temperature","properties": {"value": 57}},{"service_id": "Battery","properties": {"level": 80}}]，具体格式需要应用和设备约定, 最大长度为256KB。 最大长度：**262144**

响应参数

状态码： 200

表 1-137 响应 Body 参数

参数	参数类型	描述
request_id	String	设备属性更新ID，用于唯一标识一条属性更新，在下发更新属性时由物联网平台分配获得。
response	Object	设备上报的属性执行结果。Json格式，具体格式需要应用和设备约定。
error_code	String	属性更新异常错误码。
error_msg	String	属性更新异常错误信息。

请求示例

下发设备属性，服务id为Temperature的属性为value，值为57，服务id为Batter的属性为level，值为80。

```
PUT https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/properties

{
  "services" : [ {
    "service_id" : "Temperature",
    "properties" : {
      "value" : 57
    }
  }, {
    "service_id" : "Battery",
    "properties" : {
      "level" : 80
    }
```



```
  } ]
}
```

响应示例

状态码： 200

OK

```
{
  "request_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "response" : {
    "result_code" : 0,
    "result_desc" : "success"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

下发设备属性，服务id为Temperature的属性为value，值为57，服务id为Batter的属性为level，值为80。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdatePropertiesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdatePropertiesRequest request = new UpdatePropertiesRequest();
```



```
        DevicePropertiesRequest body = new DevicePropertiesRequest();
        body.withServices("[{\"service_id\":\"Temperature\",\"properties\":{\"value\":57}}, {\"service_id
\":\"Battery\",\"properties\":{\"level\":80}}]");
        request.withBody(body);
        try {
            UpdatePropertiesResponse response = client.updateProperties(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

下发设备属性，服务id为Temperature的属性为value，值为57，服务id为Batter的属性为level，值为80。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdatePropertiesRequest()
        request.body = DevicePropertiesRequest(
            services="[{\"service_id\":\"Temperature\",\"properties\":{\"value\":57}}, {\"service_id\":\"Battery
\",\"properties\":{\"level\":80}}]"
        )
        response = client.update_properties(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

下发设备属性，服务id为Temperature的属性为value，值为57，服务id为Batter的属性为level，值为80。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdatePropertiesRequest{}
    var servicesDevicePropertiesRequest interface{} = "[{\"service_id\":\"Temperature\",\"properties\":{\"value\":57}}, {\"service_id\":\"Battery\",\"properties\":{\"level\":80}}]"
    request.Body = &model.DevicePropertiesRequest{
        Services: &servicesDevicePropertiesRequest,
    }
    response, err := client.UpdateProperties(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.5.2 查询设备属性

功能介绍

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设备发送指令用以查询设备的实时属性，并由设备将属性查询的结果同步返回给应用服务器。 注意：此接口适用于MQTT设备，暂不支持NB-IoT设备。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/devices/{device_id}/properties

表 1-138 路径参数

参数	是否必选	参数类型	描述
device_id	是	String	**参数说明**：下发属性的设备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。
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-139 Query 参数

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID，在设备关联的产品模型中定义。

请求参数

表 1-140 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-141 响应 Body 参数

参数	参数类型	描述
request_id	String	设备属性查询ID，用于唯一标识一条属性查询，在下发查询属性时由物联网平台分配获得。
response	Object	设备上报的属性执行结果。Json格式，具体格式需要应用和设备约定。
error_code	String	属性查询异常错误码。
error_msg	String	属性查询异常错误信息。

请求示例

查询设备属性，查询指定设备的服务为Temperature的所有属性。

```
GET https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/properties?service_id=Temperature
```

响应示例

状态码： 200

OK

```
{
  "request_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "response" : {
    "services" : {
      "serviceId" : "Temperature",
      "properties" : {
        "PhV_phsA" : "1",
        "PhV_phsB" : "2"
      },
      "eventTime" : "20190606T121212Z"
    }
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListPropertiesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
```

```
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        ListPropertiesRequest request = new ListPropertiesRequest();
        request.withServiceId("<service_id>");
        try {
            ListPropertiesResponse response = client.listProperties(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListPropertiesRequest()
        request.service_id = "<service_id>"
        response = client.list_properties(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListPropertiesRequest{}
    request.ServiceId = "<service_id>"
    response, err := client.ListProperties(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden

状态码	描述
404	Not Found
500	Internal Server Error
503	Service Unavailable

错误码

请参见**错误码**。

1.4.6 AMQP 队列管理

1.4.6.1 创建 AMQP 队列

功能介绍

应用服务器可调用此接口在物联网平台创建一个AMQP队列。每个租户只能创建100个队列，若超过规格，则创建失败，若队列名称与已有的队列名称相同，则创建失败。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/amqp-queues

表 1-142 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-143 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-144 请求 Body 参数

参数	是否必选	参数类型	描述
queue_name	是	String	**参数说明**：队列名称，同一租户不允许重复。 **取值范围**：长度不低于8不超过128，只允许字母、数字、下划线（_）、连接符（-）、间隔号（.）、冒号（:）的组合。 最小长度：**8** 最大长度：**128**

响应参数

状态码： 201

表 1-145 响应 Body 参数

参数	参数类型	描述
queue_id	String	队列ID，用于唯一标识一个队列。
queue_name	String	队列名称，同一租户不允许重复。 最小长度：**8** 最大长度：**128**

参数	参数类型	描述
create_time	String	在物联网平台创建队列的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
last_modify_time	String	在物联网平台最后修改队列的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

请求示例

创建amqp队列，名称为myQueue。

```
POST https://{endpoint}/v5/iot/{project_id}/amqp-queues

{
  "queue_name" : "myQueue"
}
```

响应示例

状态码： 201

Created

```
{
  "queue_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "queue_name" : "myQueue0",
  "create_time" : "20190303T081011Z",
  "last_modify_time" : "20190303T081011Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建amqp队列，名称为myQueue。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class AddQueueSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
```



```
        // In this example, AK and SK are stored in environment variables for authentication. Before running
this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddQueueRequest request = new AddQueueRequest();
        QueueInfo body = new QueueInfo();
        body.withQueueName("myQueue");
        request.withBody(body);
        try {
            AddQueueResponse response = client.addQueue(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建amqp队列，名称为myQueue。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
```

```
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddQueueRequest()
        request.body = QueueInfo(
            queue_name="myQueue"
        )
        response = client.add_queue(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建amqp队列，名称为myQueue。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddQueueRequest{}
    request.Body = &model.QueueInfo{
        QueueName: "myQueue",
    }
    response, err := client.AddQueue(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```

```
  }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.6.2 查询 AMQP 列表

功能介绍

应用服务器可调用此接口查询物联网平台中的AMQP队列信息列表。可通过队列名称作模糊查询，支持分页。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/amqp-queues

表 1-146 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-147 Query 参数

参数	是否必选	参数类型	描述
queue_name	否	String	**参数说明**：amqp队列名称，支持模糊查询，为空查询所有的队列（当前规格单个用户最大100个队列）。 **取值范围**：长度不超过128，只允许字母、数字、下划线（_）、连接符（-）、间隔号（.）、冒号（:）的组合。 最大长度：**128**
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-148 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-149 响应 Body 参数

参数	参数类型	描述
queues	Array of **QueryQueueBase** objects	队列信息列表。
page	**Page** object	查询结果的分页信息。

表 1-150 QueryQueueBase

参数	参数类型	描述
queue_id	String	队列ID，用于唯一标识一个队列。
queue_name	String	队列名称，同一租户不允许重复。 最小长度：**8** 最大长度：**128**
create_time	String	在物联网平台创建队列的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
last_modify_time	String	在物联网平台最后修改队列的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-151 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

列报查询所有的amqp队列。

```
GET https://{endpoint}/v5/iot/{project_id}/amqp-queues
```

响应示例

状态码： 200

OK

```
{
  "queues" : [ {
    "queue_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
```

```
    "queue_name" : "myQueue0",
    "create_time" : "20190303T081011Z",
    "last_modify_time" : "20190303T081011Z"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class BatchShowQueueSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        BatchShowQueueRequest request = new BatchShowQueueRequest();
        request.withQueueName("<queue_name>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            BatchShowQueueResponse response = client.batchShowQueue(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
```



```
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = BatchShowQueueRequest()
        request.queue_name = "<queue_name>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.batch_show_queue(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
```

```
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.BatchShowQueueRequest{}
    queueNameRequest:= "<queue_name>"
    request.QueueName = &queueNameRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.BatchShowQueue(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.6.3 查询单个 AMQP 队列

功能介绍

应用服务器可调用此接口查询物联网平台中指定队列的详细信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/amqp-queues/{queue_id}

表 1-152 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
queue_id	是	String	**参数说明**：队列ID，用于唯一标识一个队列。 **取值范围**：长度36位，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-153 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。

参数	是否必选	参数类型	描述
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-154 响应 Body 参数

参数	参数类型	描述
queue_id	String	队列ID，用于唯一标识一个队列。
queue_name	String	队列名称，同一租户不允许重复。 最小长度：**8** 最大长度：**128**
create_time	String	在物联网平台创建队列的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
last_modify_time	String	在物联网平台最后修改队列的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

请求示例

查询指定队列详情。

```
GET https://{endpoint}/v5/iot/{project_id}/amqp-queues/{queue_id}
```

响应示例

状态码： 200

OK

```
{
  "queue_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "queue_name" : "myQueue0",
  "create_time" : "20190303T081011Z",
  "last_modify_time" : "20190303T081011Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowQueueSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowQueueRequest request = new ShowQueueRequest();
        try {
            ShowQueueResponse response = client.showQueue(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```



```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowQueueRequest()
        response = client.show_queue(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowQueueRequest{}
    response, err := client.ShowQueue(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.6.4 删除 AMQP 队列

功能介绍

应用服务器可调用此接口在物联网平台上删除指定AMQP队列。若当前队列正在使用，则会删除失败。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/amqp-queues/{queue_id}

表 1-155 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
queue_id	是	String	**参数说明**：队列ID，用于唯一标识一个队列。 **取值范围**：长度36位，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-156 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除指定amqp队列。

```
DELETE https://{endpoint}/v5/iot/{project_id}/amqp-queues/{queue_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteQueueSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteQueueRequest request = new DeleteQueueRequest();
        try {
            DeleteQueueResponse response = client.deleteQueue(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```

```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteQueueRequest()
        response = client.delete_queue(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```

```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteQueueRequest{}
    response, err := client.DeleteQueue(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.7 接入凭证管理

1.4.7.1 生成接入凭证

功能介绍

接入凭证是用于客户端使用AMQP等协议与平台建链的一个认证凭据。只保留一条记录，如果重复调用只会重置接入凭证，使得之前的失效。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/auth/accesscode

表 1-157 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-158 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**0** 最大长度：**36**

表 1-159 请求 Body 参数

参数	是否必选	参数类型	描述
type	否	String	**参数说明**：接入凭证类型，默认为AMQP的接入凭证类型。 **取值范围**： • [AMQP,MQTT] 最小长度：**0** 最大长度：**16**

参数	是否必选	参数类型	描述
force_disconnect	否	Boolean	**参数说明**: 是否将AMQP/MQTT连接断开 缺省值：**false**

响应参数

状态码： 201

表 1-160 响应 Body 参数

参数	参数类型	描述
access_key	String	接入名，随机生成8位字符串 最小长度：**0** 最大长度：**8**
access_code	String	接入凭证。 最小长度：**0** 最大长度：**1024**

请求示例

生成接入凭证，接入凭证类型为amqp。

```
POST https://{endpoint}/v5/iot/{project_id}/auth/accesscode

{
  "type" : "AMQP"
}
```

响应示例

状态码： 201

Created

```
{
  "access_key" : "examples",
  "access_code" : "examples"
}
```

SDK 代码示例

SDK代码示例如下。

Java

生成接入凭证，接入凭证类型为amqp。

```
package com.huaweicloud.sdk.test;
```



```
import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateAccessCodeSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateAccessCodeRequest request = new CreateAccessCodeRequest();
        CreateAccessCodeRequestBody body = new CreateAccessCodeRequestBody();
        body.withType("AMQP");
        request.withBody(body);
        try {
            CreateAccessCodeResponse response = client.createAccessCode(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

生成接入凭证，接入凭证类型为amqp。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
```

```
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateAccessCodeRequest()
        request.body = CreateAccessCodeRequestBody(
            type="AMQP"
        )
        response = client.create_access_code(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

生成接入凭证，接入凭证类型为amqp。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```

```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateAccessCodeRequest{}
    typeCreateAccessCodeRequestBody:= "AMQP"
    request.Body = &model.CreateAccessCodeRequestBody{
        Type: &typeCreateAccessCodeRequestBody,
    }
    response, err := client.CreateAccessCode(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8 数据流转规则管理

资源空间是物联网平台中各资源的顶层概念，包括设备、产品、规则等各类资源都从属于资源空间，通过资源空间，实现一个账号下各类资源的逻辑隔离。

1.4.8.1 创建规则触发条件

功能介绍

应用服务器可调用此接口在物联网平台创建一条规则触发条件。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/routing-rule/rules

表 1-161 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-162 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-163 请求 Body 参数

参数	是否必选	参数类型	描述
rule_name	否	String	**参数说明**：规则名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
description	否	String	**参数说明**：用户自定义的规则描述。 最小长度：**0** 最大长度：**256**
subject	是	**RoutingRuleSubject** object	**参数说明**：资源事件信息。
app_type	否	String	**参数说明**：租户规则的生效范围，默认GLOBAL，。 **取值范围**： • GLOBAL：生效范围为租户级。 • APP：生效范围为资源空间级。如果类型为APP，创建的规则生效范围为携带的app_id指定的资源空间，不携带app_id则创建规则生效范围为**默认资源空间**。
app_id	否	String	**参数说明**：资源空间ID。。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
select	否	String	**参数说明**：用户自定义sql select语句，最大长度2500，该参数仅供标准版和企业版用户使用。可填参数可参考帮助文档数据流转下各接口的请求参数，如notify_data.body。 最小长度：**0** 最大长度：**2500**
where	否	String	**参数说明**：用户自定义sql where语句，最大长度2500，该参数仅供标准版和企业版用户使用,可填参数可参考帮助文档数据流转下各接口的请求参数，如notify_data.body。 最小长度：**0** 最大长度：**2500**

表 1-164 RoutingRuleSubject

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：资源名称。 **取值范围**： ● device：设备。 ● device.property：设备属性。 ● device.message：设备消息。 ● device.message.status：设备消息状态。 ● device.status：设备状态。 ● batchtask：批量任务。 ● product：产品。 ● device.command.status：设备异步命令状态。 最小长度：**1** 最大长度：**50**

参数	是否必选	参数类型	描述
event	是	String	**参数说明**：资源事件。 **取值范围**：与资源有关，不同的资源，事件不同。event需要与resource关联使用，具体的“resource：event”映射关系如下： ● device：create（设备添加） ● device：delete（设备删除） ● device：update（设备更新） ● device.status：update （设备状态变更） ● device.property：report（设备属性上报） ● device.message：report（设备消息上报） ● device.message.status：update（设备消息状态变更） ● batchtask：update （批量任务状态变更） ● product：create（产品添加） ● product：delete（产品删除） ● product：update（产品更新） ● device.command.status：update（设备异步命令状态更新） 最小长度：**1** 最大长度：**50**

响应参数

状态码： 201

表 1-165 响应 Body 参数

参数	参数类型	描述
rule_id	String	规则触发条件ID，用于唯一标识一个规则触发条件，在创建规则条件时由物联网平台分配获得。

参数	参数类型	描述
rule_name	String	用户自定义的规则名称。 最小长度：**1** 最大长度：**256**
description	String	用户自定义的规则描述。 最小长度：**1** 最大长度：**256**
subject	**RoutingRuleSubject** object	资源事件信息，即资源变化事件。
app_type	String	租户规则的生效范围，取值如下： ● GLOBAL：生效范围为租户级 ● APP：生效范围为资源空间级。
app_id	String	资源空间ID
select	String	用户自定义sql select语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
where	String	用户自定义sql where语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
active	Boolean	规则条件的状态是否为激活。

表 1-166 RoutingRuleSubject

参数	参数类型	描述
resource	String	**参数说明**：资源名称。 **取值范围**： ● device：设备。 ● device.property：设备属性。 ● device.message：设备消息。 ● device.message.status：设备消息状态。 ● device.status：设备状态。 ● batchtask：批量任务。 ● product：产品。 ● device.command.status：设备异步命令状态。 最小长度：**1** 最大长度：**50**
event	String	**参数说明**：资源事件。 **取值范围**：与资源有关，不同的资源，事件不同。event需要与resource关联使用，具体的“resource：event”映射关系如下： ● device：create（设备添加） ● device：delete（设备删除） ● device：update（设备更新） ● device.status：update （设备状态变更） ● device.property：report（设备属性上报） ● device.message：report（设备消息上报） ● device.message.status：update（设备消息状态变更） ● batchtask：update （批量任务状态变更） ● product：create（产品添加） ● product：delete（产品删除） ● product：update（产品更新） ● device.command.status：update（设备异步命令状态更新） 最小长度：**1** 最大长度：**50**

请求示例

- 创建规则触发条件，触发条件为设备创建通知。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/rules

{
```



```
  "rule_name" : "rulename",
  "subject" : {
    "resource" : "device",
    "event" : "create"
  },
  "app_type" : "GLOBAL",
  "description" : "description"
}
```

- 创建规则触发条件，触发条件为属性上报。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/rules

{
  "rule_name" : "rulename",
  "subject" : {
    "resource" : "device.property",
    "event" : "report"
  },
  "app_type" : "GLOBAL",
  "description" : "description"
}
```

- 创建规则触发条件，触发条件为消息上报(根据sql进行topic筛选，基础版不支持该sql筛选能力)。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/rules

{
  "rule_name" : "rulename",
  "subject" : {
    "resource" : "device.message",
    "event" : "report"
  },
  "app_type" : "GLOBAL",
  "description" : "description",
  "select" : "notify_data.header as header,notify_data.body as body,'12345678901234abcd' as id",
  "where" : "notify_data.body.topic = '$oc/devices/646c7579a5adc915f8966e8b_8514932826827763/user/testmsg'"
}
```

响应示例

状态码： 201

Created

```
{
  "rule_id" : "5bcaddda-75bf-4623-8c8d-26175c41fcca",
  "rule_name" : "rulename",
  "description" : "description",
  "subject" : {
    "resource" : "device",
    "event" : "create"
  },
  "app_type" : "GLOBAL",
  "app_id" : "1a7ffc5cd89c44dd8265b1653d951ce0",
  "select" : "*",
  "where" : "product_id='d89c-44dd-8265-b1653d951ce0'",
  "active" : false
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建规则触发条件，触发条件为设备创建通知。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingRuleRequest request = new CreateRoutingRuleRequest();
        AddRuleReq body = new AddRuleReq();
        RoutingRuleSubject subjectbody = new RoutingRuleSubject();
        subjectbody.withResource("device")
            .withEvent("create");
        body.withAppType("GLOBAL");
        body.withSubject(subjectbody);
        body.withDescription("description");
        body.withRuleName("rulename");
        request.withBody(body);
        try {
            CreateRoutingRuleResponse response = client.createRoutingRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建规则触发条件，触发条件为属性上报。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingRuleRequest request = new CreateRoutingRuleRequest();
        AddRuleReq body = new AddRuleReq();
        RoutingRuleSubject subjectbody = new RoutingRuleSubject();
        subjectbody.withResource("device.property")
            .withEvent("report");
        body.withAppType("GLOBAL");
        body.withSubject(subjectbody);
        body.withDescription("description");
        body.withRuleName("rulename");
        request.withBody(body);
        try {
            CreateRoutingRuleResponse response = client.createRoutingRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建规则触发条件，触发条件为消息上报(根据sql进行topic筛选，基础版不支持该sql筛选能力)。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingRuleRequest request = new CreateRoutingRuleRequest();
        AddRuleReq body = new AddRuleReq();
        RoutingRuleSubject subjectbody = new RoutingRuleSubject();
        subjectbody.withResource("device.message")
            .withEvent("report");
        body.withWhere("notify_data.body.topic = '$oc/devices/646c7579a5adc915f8966e8b_8514932826827763/user/testmsg'");
        body.withSelect("notify_data.header as header,notify_data.body as body,'12345678901234abcd' as id");
        body.withAppType("GLOBAL");
        body.withSubject(subjectbody);
        body.withDescription("description");
        body.withRuleName("rulename");
        request.withBody(body);
        try {
            CreateRoutingRuleResponse response = client.createRoutingRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
```

```
        }
    }
}
```

Python

- 创建规则触发条件，触发条件为设备创建通知。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingRuleRequest()
        subjectbody = RoutingRuleSubject(
            resource="device",
            event="create"
        )
        request.body = AddRuleReq(
            app_type="GLOBAL",
            subject=subjectbody,
            description="description",
            rule_name="rulename"
        )
        response = client.create_routing_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建规则触发条件，触发条件为属性上报。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *
```



```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingRuleRequest()
        subjectbody = RoutingRuleSubject(
            resource="device.property",
            event="report"
        )
        request.body = AddRuleReq(
            app_type="GLOBAL",
            subject=subjectbody,
            description="description",
            rule_name="rulename"
        )
        response = client.create_routing_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建规则触发条件，触发条件为消息上报(根据sql进行topic筛选，基础版不支持该sql筛选能力)。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
```

```
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingRuleRequest()
        subjectbody = RoutingRuleSubject(
            resource="device.message",
            event="report"
        )
        request.body = AddRuleReq(
            where="notify_data.body.topic = '$oc/devices/
646c7579a5adc915f8966e8b_8514932826827763/user/testmsg'",
            select="notify_data.header as header,notify_data.body as body,'12345678901234abcd' as id",
            app_type="GLOBAL",
            subject=subjectbody,
            description="description",
            rule_name="rulename"
        )
        response = client.create_routing_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建规则触发条件，触发条件为设备创建通知。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())
```

```
    request := &model.CreateRoutingRuleRequest{}
    subjectbody := &model.RoutingRuleSubject{
        Resource: "device",
        Event: "create",
    }
    appTypeAddRuleReq:= "GLOBAL"
    descriptionAddRuleReq:= "description"
    ruleNameAddRuleReq:= "rulename"
    request.Body = &model.AddRuleReq{
        AppType: &appTypeAddRuleReq,
        Subject: subjectbody,
        Description: &descriptionAddRuleReq,
        RuleName: &ruleNameAddRuleReq,
    }
    response, err := client.CreateRoutingRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建规则触发条件，触发条件为属性上报。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRoutingRuleRequest{}
    subjectbody := &model.RoutingRuleSubject{
        Resource: "device.property",
        Event: "report",
    }
    appTypeAddRuleReq:= "GLOBAL"
    descriptionAddRuleReq:= "description"
    ruleNameAddRuleReq:= "rulename"
```



```
    request.Body = &model.AddRuleReq{
        AppType: &appTypeAddRuleReq,
        Subject: subjectbody,
        Description: &descriptionAddRuleReq,
        RuleName: &ruleNameAddRuleReq,
    }
    response, err := client.CreateRoutingRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建规则触发条件，触发条件为消息上报(根据sql进行topic筛选，基础版不支持该sql筛选能力)。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRoutingRuleRequest{}
    subjectbody := &model.RoutingRuleSubject{
        Resource: "device.message",
        Event: "report",
    }
    whereAddRuleReq:= "notify_data.body.topic = '$oc/devices/646c7579a5adc915f8966e8b_8514932826827763/user/testmsg'"
    selectAddRuleReq:= "notify_data.header as header,notify_data.body as body,'12345678901234abcd' as id"
    appTypeAddRuleReq:= "GLOBAL"
    descriptionAddRuleReq:= "description"
    ruleNameAddRuleReq:= "rulename"
    request.Body = &model.AddRuleReq{
        Where: &whereAddRuleReq,
        Select: &selectAddRuleReq,
```

```
        AppType: &appTypeAddRuleReq,
        Subject: subjectbody,
        Description: &descriptionAddRuleReq,
        RuleName: &ruleNameAddRuleReq,
    }
    response, err := client.CreateRoutingRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.2 查询规则条件列表

功能介绍

应用服务器可调用此接口查询物联网平台中设置的规则条件列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/rules

表 1-167 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-168 Query 参数

参数	是否必选	参数类型	描述
resource	否	String	**参数说明**：订阅的资源名称。 **取值范围**： • device：设备。 • device.property：设备属性。 • device.message：设备消息。 • device.message.status：设备消息状态。 • device.status：设备状态。 • batchtask：批量任务。 • product：产品。 • device.command.status：设备异步命令状态。 最小长度：**1** 最大长度：**50**

参数	是否必选	参数类型	描述
event	否	String	**参数说明**：订阅的资源事件。 **取值范围**：与资源有关，不同的资源，事件不同。 event需要与resource关联使用，具体的“resource：event”映射关系如下： • device：create（设备添加） • device：delete（设备删除） • device：update（设备更新） • device.status：update （设备状态变更） • device.property：report（设备属性上报） • device.message：report（设备消息上报） • device.message.status：update（设备消息状态变更） • batchtask：update （批量任务状态变更） • product：create（产品添加） • product：delete（产品删除） • product：update（产品更新） • device.command.status：update（设备异步命令状态更新）。 最小长度：**1** 最大长度：**50**
app_type	否	String	**参数说明**：租户规则的生效范围。 **取值范围**： • GLOBAL：生效范围为租户级。 • APP：生效范围为资源空间级。如果类型为APP，可携带app_id查询指定资源空间下的规则动作列表，不携带app_id则查询**默认资源空间**下的规则列表。

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，携带app_id查询指定资源空间下的规则动作列表，不携带app_id则查询**默认资源空间**下的规则动作列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
rule_name	否	String	**参数说明**：用户自定义的规则名称 最小长度：**1** 最大长度：**256**
active	否	Boolean	**参数说明**：规则条件的状态是否为激活。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。默认每页10条记录，最大设定每页50条记录。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。 - 限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-169 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-170 响应 Body 参数

参数	参数类型	描述
rules	Array of **RoutingRule** objects	规则条件信息列表。
count	Integer	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

表 1-171 RoutingRule

参数	参数类型	描述
rule_id	String	规则触发条件ID，用于唯一标识一个规则触发条件，在创建规则条件时由物联网平台分配获得。
rule_name	String	用户自定义的规则名称。 最小长度：**1** 最大长度：**256**
description	String	用户自定义的规则描述。 最小长度：**1** 最大长度：**256**
subject	**RoutingRuleSubject** object	资源事件信息，即资源变化事件。
app_type	String	租户规则的生效范围，取值如下： • GLOBAL：生效范围为租户级 • APP：生效范围为资源空间级。
app_id	String	资源空间ID
select	String	用户自定义sql select语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
where	String	用户自定义sql where语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
active	Boolean	规则条件的状态是否为激活。

表 1-172 RoutingRuleSubject

参数	参数类型	描述
resource	String	**参数说明**：资源名称。 **取值范围**： ● device：设备。 ● device.property：设备属性。 ● device.message：设备消息。 ● device.message.status：设备消息状态。 ● device.status：设备状态。 ● batchtask：批量任务。 ● product：产品。 ● device.command.status：设备异步命令状态。 最小长度：**1** 最大长度：**50**
event	String	**参数说明**：资源事件。 **取值范围**：与资源有关，不同的资源，事件不同。event需要与resource关联使用，具体的“resource：event”映射关系如下： ● device：create（设备添加） ● device：delete（设备删除） ● device：update（设备更新） ● device.status：update （设备状态变更） ● device.property：report（设备属性上报） ● device.message：report（设备消息上报） ● device.message.status：update（设备消息状态变更） ● batchtask：update （批量任务状态变更） ● product：create（产品添加） ● product：delete（产品删除） ● product：update（产品更新） ● device.command.status：update（设备异步命令状态更新） 最小长度：**1** 最大长度：**50**

请求示例

列表查询数据流转规则。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/rules
```


响应示例

状态码： 200

Successful response

```
{
  "rules" : [ {
    "rule_id" : "5bcaddda-75bf-4623-8c8d-26175c41fcca",
    "app_type" : "GLOBAL",
    "select" : "*",
    "rule_name" : "rulename",
    "subject" : {
      "resource" : "device",
      "event" : "create"
    },
    "description" : "description",
    "active" : true,
    "where" : "product_id='d89c-44dd-8265-b1653d951ce0'",
    "app_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0"
  } ],
  "count" : 10,
  "marker" : "5c90fa7d3c4e4405e8525079"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListRoutingRulesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
```

```
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListRoutingRulesRequest request = new ListRoutingRulesRequest();
        request.withResource("<resource>");
        request.withEvent("<event>");
        request.withAppType("<app_type>");
        request.withAppId("<app_id>");
        request.withRuleName("<rule_name>");
        request.withActive(<active>);
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListRoutingRulesResponse response = client.listRoutingRules(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListRoutingRulesRequest()
        request.resource = "<resource>"
        request.event = "<event>"
        request.app_type = "<app_type>"
        request.app_id = "<app_id>"
        request.rule_name = "<rule_name>"
```



```
        request.active = <Active>
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_routing_rules(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListRoutingRulesRequest{}
    resourceRequest:= "<resource>"
    request.Resource = &resourceRequest
    eventRequest:= "<event>"
    request.Event = &eventRequest
    appTypeRequest:= "<app_type>"
    request.AppType = &appTypeRequest
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    ruleNameRequest:= "<rule_name>"
    request.RuleName = &ruleNameRequest
    activeRequest:= <active>
    request.Active = &activeRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
```

```
    request.Offset = &offsetRequest
    response, err := client.ListRoutingRules(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Successful response
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.3 查询规则条件

功能介绍

应用服务器可调用此接口查询物联网平台中指定规则条件的配置信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/rules/{rule_id}

表 1-173 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

参数	是否必选	参数类型	描述
rule_id	是	String	**参数说明**：规则条件ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-174 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-175 响应 Body 参数

参数	参数类型	描述
rule_id	String	规则触发条件ID，用于唯一标识一个规则触发条件，在创建规则条件时由物联网平台分配获得。
rule_name	String	用户自定义的规则名称。 最小长度：**1** 最大长度：**256**
description	String	用户自定义的规则描述。 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
subject	RoutingRuleSubject object	资源事件信息，即资源变化事件。
app_type	String	租户规则的生效范围，取值如下： • GLOBAL：生效范围为租户级 • APP：生效范围为资源空间级。
app_id	String	资源空间ID
select	String	用户自定义sql select语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
where	String	用户自定义sql where语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
active	Boolean	规则条件的状态是否为激活。

表 1-176 RoutingRuleSubject

参数	参数类型	描述
resource	String	**参数说明**：资源名称。 **取值范围**： • device：设备。 • device.property：设备属性。 • device.message：设备消息。 • device.message.status：设备消息状态。 • device.status：设备状态。 • batchtask：批量任务。 • product：产品。 • device.command.status：设备异步命令状态。 最小长度：**1** 最大长度：**50**

参数	参数类型	描述
event	String	**参数说明**：资源事件。 **取值范围**：与资源有关，不同的资源，事件不同。event需要与resource关联使用，具体的“resource：event”映射关系如下： ● device：create（设备添加） ● device：delete（设备删除） ● device：update（设备更新） ● device.status：update （设备状态变更） ● device.property：report（设备属性上报） ● device.message：report（设备消息上报） ● device.message.status：update（设备消息状态变更） ● batchtask：update （批量任务状态变更） ● product：create（产品添加） ● product：delete（产品删除） ● product：update（产品更新） ● device.command.status：update（设备异步命令状态更新） 最小长度：**1** 最大长度：**50**

请求示例

查询指定规则详情。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/rules/{rule_id}
```

响应示例

状态码： 200

OK

```
{
  "rule_id" : "5bcaddda-75bf-4623-8c8d-26175c41fcca",
  "rule_name" : "rulename",
  "description" : "description",
  "subject" : {
    "resource" : "device",
    "event" : "create"
  },
  "app_type" : "GLOBAL",
  "app_id" : "1a7ffc5cd89c44dd8265b1653d951ce0",
  "select" : "*",
  "where" : "product_id='d89c-44dd-8265-b1653d951ce0'",
  "active" : false
}
```


SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowRoutingRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowRoutingRuleRequest request = new ShowRoutingRuleRequest();
        try {
            ShowRoutingRuleResponse response = client.showRoutingRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```

```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowRoutingRuleRequest()
        response = client.show_routing_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```

```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowRoutingRuleRequest{}
    response, err := client.ShowRoutingRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.4 修改规则触发条件

功能介绍

应用服务器可调用此接口修改物联网平台中指定规则条件的配置参数。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/routing-rule/rules/{rule_id}

表 1-177 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
rule_id	是	String	**参数说明**：规则条件ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-178 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-179 请求 Body 参数

参数	是否必选	参数类型	描述
rule_name	否	String	**参数说明**：规则名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
description	否	String	**参数说明**：用户自定义的规则描述。 最小长度：**0** 最大长度：**256**
select	否	String	**参数说明**：用户自定义sql select语句，最大长度2500，更新sql时，select跟where必须同时传参，如果需要清除该参数的值，输入空字符串，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
where	否	String	**参数说明**：用户自定义sql where语句，最大长度2500，更新操作时，select跟where必须同时传参，如果需要清除该参数的值，输入空字符串，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
active	否	Boolean	**参数说明**：修改规则条件的状态是否为激活。

响应参数

状态码： 200

表 1-180 响应 Body 参数

参数	参数类型	描述
rule_id	String	规则触发条件ID，用于唯一标识一个规则触发条件，在创建规则条件时由物联网平台分配获得。
rule_name	String	用户自定义的规则名称。 最小长度：**1** 最大长度：**256**
description	String	用户自定义的规则描述。 最小长度：**1** 最大长度：**256**
subject	**RoutingRuleSubject** object	资源事件信息，即资源变化事件。

参数	参数类型	描述
app_type	String	租户规则的生效范围，取值如下： ● GLOBAL：生效范围为租户级 ● APP：生效范围为资源空间级。
app_id	String	资源空间ID
select	String	用户自定义sql select语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
where	String	用户自定义sql where语句，最大长度2500，该参数仅供标准版和企业版用户使用。 最小长度：**0** 最大长度：**2500**
active	Boolean	规则条件的状态是否为激活。

表 1-181 RoutingRuleSubject

参数	参数类型	描述
resource	String	**参数说明**：资源名称。 **取值范围**： ● device：设备。 ● device.property：设备属性。 ● device.message：设备消息。 ● device.message.status：设备消息状态。 ● device.status：设备状态。 ● batchtask：批量任务。 ● product：产品。 ● device.command.status：设备异步命令状态。 最小长度：**1** 最大长度：**50**

参数	参数类型	描述
event	String	**参数说明**：资源事件。 **取值范围**：与资源有关，不同的资源，事件不同。event需要与resource关联使用，具体的“resource：event”映射关系如下： ● device：create（设备添加） ● device：delete（设备删除） ● device：update（设备更新） ● device.status：update （设备状态变更） ● device.property：report（设备属性上报） ● device.message：report（设备消息上报） ● device.message.status：update（设备消息状态变更） ● batchtask：update （批量任务状态变更） ● product：create（产品添加） ● product：delete（产品删除） ● product：update（产品更新） ● device.command.status：update（设备异步命令状态更新） 最小长度：**1** 最大长度：**50**

请求示例

修改指定规则触发条件,指定条件为产品为d89c-44dd-8265-b1653d951ce0。

```
PUT https://{endpoint}/v5/iot/{project_id}/routing-rule/rules/{rule_id}

{
  "rule_name" : "rulename",
  "description" : "description",
  "select" : "*",
  "where" : "product_id='d89c-44dd-8265-b1653d951ce0'",
  "active" : true
}
```

响应示例

状态码： 200

OK

```
{
  "rule_id" : "5bcaddda-75bf-4623-8c8d-26175c41fcca",
  "rule_name" : "rulename",
  "description" : "description",
  "subject" : {
    "resource" : "device",
    "event" : "create"
  },
```



```
  "app_type" : "GLOBAL",
  "app_id" : "1a7ffc5cd89c44dd8265b1653d951ce0",
  "select" : "*",
  "where" : "product_id='d89c-44dd-8265-b1653d951ce0'",
  "active" : false
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改指定规则触发条件,指定条件为产品为d89c-44dd-8265-b1653d951ce0。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateRoutingRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateRoutingRuleRequest request = new UpdateRoutingRuleRequest();
        UpdateRuleReq body = new UpdateRuleReq();
        body.withActive(true);
        body.withWhere("product_id='d89c-44dd-8265-b1653d951ce0'");
        body.withSelect("*");
        body.withDescription("description");
        body.withRuleName("rulename");
        request.withBody(body);
        try {
            UpdateRoutingRuleResponse response = client.updateRoutingRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
```

```
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改指定规则触发条件,指定条件为产品为d89c-44dd-8265-b1653d951ce0。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateRoutingRuleRequest()
        request.body = UpdateRuleReq(
            active=True,
            where="product_id='d89c-44dd-8265-b1653d951ce0'",
            select="*",
            description="description",
            rule_name="rulename"
        )
        response = client.update_routing_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改指定规则触发条件,指定条件为产品为d89c-44dd-8265-b1653d951ce0。

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateRoutingRuleRequest{}
    activeUpdateRuleReq:= true
    whereUpdateRuleReq:= "product_id='d89c-44dd-8265-b1653d951ce0'"
    selectUpdateRuleReq:= "*"
    descriptionUpdateRuleReq:= "description"
    ruleNameUpdateRuleReq:= "rulename"
    request.Body = &model.UpdateRuleReq{
        Active: &activeUpdateRuleReq,
        Where: &whereUpdateRuleReq,
        Select: &selectUpdateRuleReq,
        Description: &descriptionUpdateRuleReq,
        RuleName: &ruleNameUpdateRuleReq,
    }
    response, err := client.UpdateRoutingRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK

状态码	描述
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.5 删除规则触发条件

功能介绍

应用服务器可调用此接口删除物联网平台中的指定规则条件。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/routing-rule/rules/{rule_id}

表 1-182 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
rule_id	是	String	**参数说明**：规则条件ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-183 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

无

请求示例

删除指定规则触发条件。

```
DELETE https://{endpoint}/v5/iot/{project_id}/routing-rule/rules/{rule_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```

```
public class DeleteRoutingRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteRoutingRuleRequest request = new DeleteRoutingRuleRequest();
        try {
            DeleteRoutingRuleResponse response = client.deleteRoutingRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteRoutingRuleRequest()
        response = client.delete_routing_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteRoutingRuleRequest{}
    response, err := client.DeleteRoutingRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.6 创建规则动作

功能介绍

应用服务器可调用此接口在物联网平台创建一条规则动作。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/routing-rule/actions

表 1-184 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-185 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-186 请求 Body 参数

参数	是否必选	参数类型	描述
rule_id	是	String	**参数说明**：规则触发条件ID，用于唯一标识一条规则触发条件，在创建规则时由物联网平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

参数	是否必选	参数类型	描述
channel	是	String	**参数说明**：规则动作的类型。 **取值范围**： • HTTP_FORWARDING：HTTP服务消息类型。 • DIS_FORWARDING：转发DIS服务消息类型。 • OBS_FORWARDING：转发OBS服务消息类型。 • AMQP_FORWARDING：转发AMQP服务消息类型。 • DMS_KAFKA_FORWARDING：转发kafka消息类型。 • ROMA_FORWARDING：转发Roma消息类型。（仅企业版支持） • INFLUXDB_FORWARDING：转发InfluxDB消息类型。（仅标准版和企业版支持） • MYSQL_FORWARDING：转发MySQL消息类型。（仅标准版和企业版支持） • FUNCTIONGRAPH_FORWARDING：转发FunctionGraph消息类型。（仅标准版和企业版支持） • MRS_KAFKA_FORWARDING：转发MRS_KAFKA消息类型。（仅企业版支持） • DMS_ROCKETMQ_FORWARDING：转发RocketMQ消息类型。（仅标准版和企业版支持）
channel_detail	是	**ChannelDetail** object	**参数说明**：通道参数。

表 1-187 ChannelDetail

参数	是否必选	参数类型	描述
http_forwarding	否	**HttpForwarding** object	**参数说明**：http服务器转发消息内容。当channel为HTTP_FORWARDING时，必填。

参数	是否必选	参数类型	描述
dis_forwarding	否	**DisForwarding** object	**参数说明**：转发DIS服务消息内容。当channel为DIS_FORWARDING时，必填。
obs_forwarding	否	**ObsForwarding** object	**参数说明**：转发OBS服务消息内容。当channel为OBS_FORWARDING时，必填。
amqp_forwarding	否	**AmqpForwarding** object	**参数说明**：转发AMQP服务消息内容。当channel为AMQP_FORWARDING时，必填。
dms_kafka_forwarding	否	**DmsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为DMS_KAFKA_FORWARDING时，必填。
roma_forwarding	否	**RomaForwarding** object	**参数说明**：转发Roma消息内容。当channel为ROMA_FORWARDING时,必填。（仅企业版支持）
mysql_forwarding	否	**MysqlForwarding** object	**参数说明**：转发MySQL消息内容。当channel为MYSQL_FORWARDING时,必填。
influxdb_forwarding	否	**InfluxDBForwarding** object	**参数说明**：转发influxdb的配置参数。当channel为INFLUXDB_FORWARDING时，必填。（仅标准版和企业版支持）
functiongraph_forwarding	否	**FunctionGraphForwarding** object	**参数说明**：转发云服务函数服务消息内容。当channel为FUNCTIONGRAPH_FORWARDING时，必填。（仅标准版和企业版支持）
mrs_kafka_forwarding	否	**MrsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为MRS_KAFKA_FORWARDING时，必填。（仅企业版支持）
dms_rocketmq_forwarding	否	**DmsRocketMQForwarding** object	**参数说明**：转发rocketmq消息内容。当channel为DMS_ROCKETMQ_FORWARDING时，必填。（仅标准版和企业版支持）

表 1-188 HttpForwarding

参数	是否必选	参数类型	描述
url	是	String	**参数说明**：用于接收满足规则条件数据的http服务器地址。HTTP为非数据加密传输模式，此模式下数据传输不安全， 建议使用更安全的HTTPS方式 最小长度：**1** 最大长度：**256**
cert_id	否	String	**参数说明**：证书id，请参见**加载推送证书第3步**获取证书ID 最小长度：**1** 最大长度：**64**
cn_name	否	String	**参数说明**：当sni_enable为true时，此字段需要填写，内容为将要请求的服务端证书的域名,举例:domain:8443;当sni_enbale为false时，此字段默认不填写。 最小长度：**1** 最大长度：**64**
sni_enable	否	Boolean	**参数说明**：需要https服务端和客户端都支持此功能，默认为false，设成true表明Https的客户端在发起请求时，需要携带cn_name；https服务端根据cn_name返回对应的证书；设为false可关闭此功能。
signature_enable	否	Boolean	**参数说明**：是否启用签名。填写token时， 该参数必须为true，token才可以生效，否则token不生效。推荐设置成true，使用token签名验证消息是否来自平台。
token	否	String	**参数说明**：用作生成签名的Token，客户端可以使用该token按照规则生成签名并与推送消息中携带的签名做对比，从而验证安全性。**取值范围**: 长度不超过32， 不小于3， 只允许字母、数字的组合。请参见**HTTP/HTTPS推送基于Token认证物联网平台** 最小长度：**3** 最大长度：**32**

表 1-189 DisForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：DIS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	是	String	**参数说明**：DIS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
stream_name	否	String	**参数说明**：DIS服务对应的通道名称，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**
stream_id	否	String	**参数说明**：DIS服务对应的通道ID，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

表 1-190 ObsForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：OBS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	是	String	**参数说明**：OBS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
bucket_name	是	String	**参数说明**：OBS服务对应的桶名称 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
location	否	String	**参数说明**：OBS服务对应桶的区域 最小长度：**1** 最大长度：**256**
file_path	否	String	**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(/) **取值范围**: 英文字母(a-zA-Z)、数字(0-9)、下划线(_)、中划线(-)、斜杠(/)和大括号({})，最大字符长度256个字符。其中大括号只能用于对应模板参数。 **模板参数**: ● {YYYY} 年 ● {MM} 月 ● {DD} 日 ● {HH} 小时 ● {appId} 应用ID ● {deviceId} 设备ID 例如:自定义目录结构为{YYYY}/{MM}/{DD}/{HH},则会在转发数据时，根据当前时间往对应的目录结构2021>08>11>09下生成对应的数据。

表 1-191 AmqpForwarding

参数	是否必选	参数类型	描述
queue_name	是	String	**参数说明**：用于接收满足规则条件数据的amqp queue。 最小长度：**8** 最大长度：**256**

表 1-192 DmsKafkaForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：Kafka服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	是	String	**参数说明**：Kafka服务对应的projectId信息 最小长度：**1** 最大长度：**256**
addresses	是	Array of **SupportPrivateLinkNetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	是	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	否	String	**参数说明**：转发kafka关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	否	String	**参数说明**：转发kafka关联的密码信息。 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
mechanism	否	String	**参数说明**：转发kafka关联的SASL认证机制。 **取值范围**： ● PAAS：明文传输，此模式下为非数据加密传输模式，数据传输不安全，建议您使用更安全的数据加密模式。 ● PLAIN：SASL/PLAIN模式。需要填写对应的用户名密码信息。一种简单的用户名密码校验机制，在SASL_PLAINTEXT场景下，不建议使用。 ● SCRAM-SHA-512：SASL/SCRAM-SHA-512模式。需要填写对应的用户名密码信息。采用哈希算法对用户名与密码生成凭证，进行身份校验的安全认证机制，比PLAIN机制安全性更高。
security_protocol	否	String	**参数说明**：kafka传输安全协议，此字段不填默认为SASL_SSL。当mechanism为PAAS或不填时，该字段不生效。 **取值范围**： ● SASL_SSL：采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT：明文传输，支持账号密码认证，性能更好，建议mechanism使用SCRAM-SHA-512机制。

表 1-193 SupportPrivateLinkNetAddress

参数	是否必选	参数类型	描述
ip	否	String	**参数说明**：服务的对应IP
port	否	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	否	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-194 RomaForwarding

参数	是否必选	参数类型	描述
addresses	是	Array of **NetAddress** objects	**参数说明**：转发roma消息对应的地址列表
topic	是	String	**参数说明**：转发roma消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	是	String	**参数说明**：转发roma关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：转发roma关联的密码信息。 最大长度：**256**

表 1-195 MysqlForwarding

参数	是否必选	参数类型	描述
address	是	**NetAddress** object	转发roma消息对应的地址列表
db_name	是	String	**参数说明**：连接MYSQL数据库的库名。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合。
username	是	String	**参数说明**：连接MYSQL数据库的用户名 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：连接MYSQL数据库的密码 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
enable_ssl	否	Boolean	**参数说明**：客户端是否使用SSL连接服务端，默认为true， 若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您使用SSL模式。 缺省值：**true**
table_name	是	String	**参数说明**：MYSQL数据库的表名 最小长度：**1** 最大长度：**64**
column_mappings	是	Array of **ColumnMapping** objects	**参数说明**：MYSQL数据库的列和流转数据的对应关系列表，最多支持32条映射关系。

表 1-196 InfluxDBForwarding

参数	是否必选	参数类型	描述
address	是	**NetAddress** object	**参数说明**：转发InfluxDB消息对应的地址
db_name	是	String	**参数说明**：连接InfluxDB数据库的库名,不存在会自动创建 最小长度：**1** 最大长度：**64**
username	是	String	**参数说明**：连接InfluxDB数据库的用户名 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：连接InfluxDB数据库的密码 最小长度：**1** 最大长度：**256**
measurement	是	String	**参数说明**：InfluxDB数据库的measurement,不存在会自动创建 最小长度：**1** 最大长度：**64**
column_mappings	是	Array of **ColumnMapping** objects	**参数说明**：InfluxDB数据库和流转数据的对应关系列表，最多支持32条映射关系。

表 1-197 NetAddress

参数	是否必选	参数类型	描述
ip	否	String	**参数说明**：服务的对应IP
port	否	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	否	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-198 ColumnMapping

参数	是否必选	参数类型	描述
column_name	是	String	**参数说明**：数据库的列名 最小长度：**1** 最大长度：**256**
json_key	是	String	**参数说明**：流转数据的属性名 最小长度：**1** 最大长度：**256**

表 1-199 FunctionGraphForwarding

参数	是否必选	参数类型	描述
func_urn	是	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数。 最小长度：**0** 最大长度：**65535**
func_name	是	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**

表 1-200 MrsKafkaForwarding

参数	是否必选	参数类型	描述
addresses	是	Array of **NetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	是	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
kerberos_authentication	否	Boolean	是否Kerberos认证，默认为false。 缺省值：**false**

表 1-201 DmsRocketMQForwarding

参数	是否必选	参数类型	描述
addresses	是	Array of **NetAddress** objects	**参数说明**：转发rocketMQ消息对应的地址列表
topic	是	String	**参数说明**：转发rocketMQ消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	是	String	**参数说明**：转发rocketMQ关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：转发rocketMQ关联的密码信息。 最小长度：**1** 最大长度：**256**
enable_ssl	否	Boolean	是否开启SSL，默认为true。若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您开启SSL。 缺省值：**true**

响应参数

状态码： 201

表 1-202 响应 Body 参数

参数	参数类型	描述
action_id	String	规则动作ID，用于唯一标识一条规则动作，在创建规则动作时由物联网平台分配获得，创建时无需携带，由平台统一分配唯一的action_id。
rule_id	String	规则动作对应的的规则触发条件ID。
app_id	String	资源空间ID。
channel	String	规则动作的类型，取值范围： • HTTP_FORWARDING：HTTP服务消息类型。 • DIS_FORWARDING：转发DIS服务消息类型。 • OBS_FORWARDING：转发OBS服务消息类型。 • AMQP_FORWARDING：转发AMQP服务消息类型。 • DMS_KAFKA_FORWARDING：转发kafka消息类型。 • ROMA_FORWARDING: 转发Roma消息类型。（仅企业版支持） • INFLUXDB_FORWARDING：转发InfluxDB消息类型。（仅标准版和企业版支持） • MYSQL_FORWARDING：转发MySQL消息类型。（仅标准版和企业版支持） • FUNCTIONGRAPH_FORWARDING：转发FunctionGraph消息类型。（仅标准版和企业版支持） • MRS_KAFKA_FORWARDING：转发MRS_KAFKA消息类型。（仅企业版支持） • DMS_ROCKETMQ_FORWARDING：转发RocketMQ消息类型。（仅标准版和企业版支持）
channel_detail	**ChannelDetail** object	通道配置信息。

表 1-203 ChannelDetail

参数	参数类型	描述
http_forwarding	**HttpForwarding** object	**参数说明**：http服务器转发消息内容。当channel为HTTP_FORWARDING时，必填。
dis_forwarding	**DisForwarding** object	**参数说明**：转发DIS服务消息内容。当channel为DIS_FORWARDING时，必填。

参数	参数类型	描述
obs_forwarding	**ObsForwarding** object	**参数说明**：转发OBS服务消息内容。当channel为OBS_FORWARDING时，必填。
amqp_forwarding	**AmqpForwarding** object	**参数说明**：转发AMQP服务消息内容。当channel为AMQP_FORWARDING时，必填。
dms_kafka_forwarding	**DmsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为DMS_KAFKA_FORWARDING时，必填。
roma_forwarding	**RomaForwarding** object	**参数说明**：转发Roma消息内容。当channel为ROMA_FORWARDING时,必填。（仅企业版支持）
mysql_forwarding	**MysqlForwarding** object	**参数说明**：转发MySQL消息内容。当channel为MYSQL_FORWARDING时,必填。
influxdb_forwarding	**InfluxDBForwarding** object	**参数说明**：转发influxdb的配置参数。当channel为INFLUXDB_FORWARDING时，必填。（仅标准版和企业版支持）
functiongraph_forwarding	**FunctionGraphForwarding** object	**参数说明**：转发云服务函数服务消息内容。当channel为FUNCTIONGRAPH_FORWARDING时，必填。（仅标准版和企业版支持）
mrs_kafka_forwarding	**MrsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为MRS_KAFKA_FORWARDING时，必填。（仅企业版支持）
dms_rocketmq_forwarding	**DmsRocketMQForwarding** object	**参数说明**：转发rocketmq消息内容。当channel为DMS_ROCKETMQ_FORWARDING时，必填。（仅标准版和企业版支持）

表 1-204 HttpForwarding

参数	参数类型	描述
url	String	**参数说明**：用于接收满足规则条件数据的http服务器地址。HTTP为非数据加密传输模式，此模式下数据传输不安全， 建议使用更安全的HTTPS方式 最小长度：**1** 最大长度：**256**
cert_id	String	**参数说明**：证书id，请参见**加载推送证书第3步**获取证书ID 最小长度：**1** 最大长度：**64**

参数	参数类型	描述
cn_name	String	**参数说明**：当sni_enable为true时，此字段需要填写，内容为将要请求的服务端证书的域名,举例:domain:8443;当sni_enbale为false时，此字段默认不填写。 最小长度：**1** 最大长度：**64**
sni_enable	Boolean	**参数说明**：需要https服务端和客户端都支持此功能，默认为false，设成true表明Https的客户端在发起请求时，需要携带cn_name；https服务端根据cn_name返回对应的证书；设为false可关闭此功能。
signature_enable	Boolean	**参数说明**：是否启用签名。填写token时， 该参数必须为true， token才可以生效，否则token不生效。推荐设置成true，使用token签名验证消息是否来自平台。
token	String	**参数说明**：用作生成签名的Token，客户端可以使用该token按照规则生成签名并与推送消息中携带的签名做对比， 从而验证安全性。**取值范围**:长度不超过32， 不小于3， 只允许字母、数字的组合。请参见**HTTP/HTTPS推送基于Token认证 物联网平台** 最小长度：**3** 最大长度：**32**

表 1-205 DisForwarding

参数	参数类型	描述
region_name	String	**参数说明**：DIS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：DIS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
stream_name	String	**参数说明**：DIS服务对应的通道名称，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
stream_id	String	**参数说明**：DIS服务对应的通道ID，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

表 1-206 ObsForwarding

参数	参数类型	描述
region_name	String	**参数说明**：OBS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：OBS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
bucket_name	String	**参数说明**：OBS服务对应的桶名称 最小长度：**1** 最大长度：**256**
location	String	**参数说明**：OBS服务对应桶的区域 最小长度：**1** 最大长度：**256**
file_path	String	**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(/) **取值范围**: 英文字母(a-zA-Z)、数字(0-9)、下划线(_)、中划线(-)、斜杠(/)和大括号({})，最大字符长度256个字符。其中大括号只能用于对应模板参数。 **模板参数**: ● {YYYY} 年 ● {MM} 月 ● {DD} 日 ● {HH} 小时 ● {appId} 应用ID ● {deviceId} 设备ID 例如:自定义目录结构为{YYYY}/{MM}/{DD}/{HH},则会在转发数据时，根据当前时间往对应的目录结构2021>08>11>09下生成对应的数据。

表 1-207 AmqpForwarding

参数	参数类型	描述
queue_name	String	**参数说明**：用于接收满足规则条件数据的amqp queue。 最小长度：**8** 最大长度：**256**

表 1-208 DmsKafkaForwarding

参数	参数类型	描述
region_name	String	**参数说明**：Kafka服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：Kafka服务对应的projectId信息 最小长度：**1** 最大长度：**256**
addresses	Array of **SupportPrivateLinkNetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发kafka关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发kafka关联的密码信息。 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
mechanism	String	**参数说明**：转发kafka关联的SASL认证机制。 **取值范围**： ● PAAS：明文传输，此模式下为非数据加密传输模式，数据传输不安全，建议您使用更安全的数据加密模式。 ● PLAIN：SASL/PLAIN模式。需要填写对应的用户名密码信息。一种简单的用户名密码校验机制，在SASL_PLAINTEXT场景下，不建议使用。 ● SCRAM-SHA-512：SASL/SCRAM-SHA-512模式。需要填写对应的用户名密码信息。采用哈希算法对用户名与密码生成凭证，进行身份校验的安全认证机制，比PLAIN机制安全性更高。
security_protocol	String	**参数说明**：kafka传输安全协议，此字段不填默认为SASL_SSL。当mechanism为PAAS或不填时，该字段不生效。 **取值范围**： ● SASL_SSL：采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT：明文传输，支持账号密码认证，性能更好，建议mechanism使用SCRAM-SHA-512机制。

表 1-209 SupportPrivateLinkNetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-210 RomaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发roma消息对应的地址列表

参数	参数类型	描述
topic	String	**参数说明**：转发roma消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发roma关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发roma关联的密码信息。 最大长度：**256**

表 1-211 MysqlForwarding

参数	参数类型	描述
address	**NetAddress** object	转发roma消息对应的地址列表
db_name	String	**参数说明**：连接MYSQL数据库的库名。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合。
username	String	**参数说明**：连接MYSQL数据库的用户名 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：连接MYSQL数据库的密码 最小长度：**1** 最大长度：**256**
enable_ssl	Boolean	**参数说明**：客户端是否使用SSL连接服务端，默认为true， 若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您使用SSL模式。 缺省值：**true**
table_name	String	**参数说明**：MYSQL数据库的表名 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：MYSQL数据库的列和流转数据的对应关系列表，最多支持32条映射关系。

表 1-212 InfluxDBForwarding

参数	参数类型	描述
address	**NetAddress** object	**参数说明**：转发InfluxDB消息对应的地址
db_name	String	**参数说明**：连接InfluxDB数据库的库名,不存在会自动创建 最小长度：**1** 最大长度：**64**
username	String	**参数说明**：连接InfluxDB数据库的用户名 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：连接InfluxDB数据库的密码 最小长度：**1** 最大长度：**256**
measurement	String	**参数说明**：InfluxDB数据库的measurement,不存在会自动创建 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：InfluxDB数据库和流转数据的对应关系列表，最多支持32条映射关系。

表 1-213 NetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-214 ColumnMapping

参数	参数类型	描述
column_name	String	**参数说明**：数据库的列名 最小长度：**1** 最大长度：**256**
json_key	String	**参数说明**：流转数据的属性名 最小长度：**1** 最大长度：**256**

表 1-215 FunctionGraphForwarding

参数	参数类型	描述
func_urn	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数。 最小长度：**0** 最大长度：**65535**
func_name	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**

表 1-216 MrsKafkaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
kerberos_authentication	Boolean	是否Kerberos认证，默认为false。 缺省值：**false**

表 1-217 DmsRocketMQForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发rocketMQ消息对应的地址列表
topic	String	**参数说明**：转发rocketMQ消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发rocketMQ关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发rocketMQ关联的密码信息。 最小长度：**1** 最大长度：**256**
enable_ssl	Boolean	是否开启SSL，默认为true。若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您开启SSL。 缺省值：**true**

请求示例

- 创建规则动作，推送至http服务器。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/actions

{
  "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "channel" : "HTTP_FORWARDING",
  "channel_detail" : {
    "http_forwarding" : {
      "url" : "http://host:port/callbackurltest"
    }
  }
}
```

- 创建规则动作，推送至obs。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/actions

{
  "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "channel" : "OBS_FORWARDING",
  "channel_detail" : {
    "obs_forwarding" : {
      "file_path" : "yourPath",
      "project_id" : "yourProjectId",
      "bucket_name" : "yourBucket_name",
      "region_name" : "yourRegion"
    }
  }
}
```

- 创建规则动作，推送至amqp队列。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/actions

{
  "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "channel" : "AMQP_FORWARDING",
  "channel_detail" : {
    "amqp_forwarding" : {
      "queue_name" : "yourQueueName"
    }
  }
}
```

- 创建规则动作，推送至mysql数据库。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/actions

{
  "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "channel" : "MYSQL_FORWARDING",
  "channel_detail" : {
    "mysql_forwarding" : {
      "address" : {
        "ip" : "yourIp",
        "port" : 3306
      },
      "username" : "userName",
      "password" : "passworld",
      "db_name" : "yourDBName",
      "table_name" : "yourTableName",
      "enable_ssl" : true,
      "column_mappings" : [ {
        "column_name" : "serviceId",
        "json_key" : "notify_data.body.services[0].service_id"
      } ]
    }
  }
}
```

响应示例

状态码： 201

Created

```
{
  "action_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce1",
  "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "app_id" : "1a7ffc5cd89c44dd8265b1653d951ce0",
  "channel" : "HTTP_FORWARDING",
  "channel_detail" : {
    "http_forwarding" : {
      "url" : "http://host:port/callbackurltest"
    }
  }
}
```

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized

状态码	描述
403	Forbidden
404	not found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.7 查询规则动作列表

功能介绍

应用服务器可调用此接口查询物联网平台中设置的规则动作列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/actions

表 1-218 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-219 Query 参数

参数	是否必选	参数类型	描述
rule_id	否	String	**参数说明**：规则触发条件ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

参数	是否必选	参数类型	描述
channel	否	String	**参数说明**：规则动作的类型。 **取值范围**： • HTTP_FORWARDING：HTTP服务消息类型。 • DIS_FORWARDING：转发DIS服务消息类型。 • OBS_FORWARDING：转发OBS服务消息类型。 • AMQP_FORWARDING：转发AMQP服务消息类型。 • DMS_KAFKA_FORWARDING：转发kafka消息类型。 • ROMA_FORWARDING：转发Roma消息类型。（仅企业版支持） • INFLUXDB_FORWARDING：转发InfluxDB消息类型。（仅标准版和企业版支持） • MYSQL_FORWARDING：转发MySQL消息类型。（仅标准版和企业版支持） • FUNCTIONGRAPH_FORWARDING：转发FunctionGraph消息类型。（仅标准版和企业版支持） • MRS_KAFKA_FORWARDING：转发MRS_KAFKA消息类型。（仅企业版支持） • DMS_ROCKETMQ_FORWARDING：转发RocketMQ消息类型。（仅标准版和企业版支持）
app_type	否	String	**参数说明**：租户规则的生效范围。 **取值范围**： • GLOBAL：生效范围为租户级。 • APP：生效范围为资源空间级。如果类型为APP，可携带app_id查询指定资源空间下的规则动作列表，不携带app_id则查询**默认资源空间**下的规则动作列表。

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，rule_id不携带且app_type为APP时，该参数生效，可携带app_id查询指定资源空间下的规则动作列表，不携带app_id则查询**默认资源空间**下的规则动作列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。默认每页10条记录，最大设定每页50条记录。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。 - 限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-220 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-221 响应 Body 参数

参数	参数类型	描述
actions	Array of **RoutingRuleAction** objects	规则动作信息列表。
count	Integer	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

表 1-222 RoutingRuleAction

参数	参数类型	描述
action_id	String	规则动作ID，用于唯一标识一条规则动作，在创建规则动作时由物联网平台分配获得，创建时无需携带，由平台统一分配唯一的action_id。
rule_id	String	规则动作对应的的规则触发条件ID。
app_id	String	资源空间ID。
channel	String	规则动作的类型，取值范围： • HTTP_FORWARDING：HTTP服务消息类型。 • DIS_FORWARDING：转发DIS服务消息类型。 • OBS_FORWARDING：转发OBS服务消息类型。 • AMQP_FORWARDING：转发AMQP服务消息类型。 • DMS_KAFKA_FORWARDING：转发kafka消息类型。 • ROMA_FORWARDING: 转发Roma消息类型。（仅企业版支持） • INFLUXDB_FORWARDING：转发InfluxDB消息类型。（仅标准版和企业版支持） • MYSQL_FORWARDING：转发MySQL消息类型。（仅标准版和企业版支持） • FUNCTIONGRAPH_FORWARDING：转发FunctionGraph消息类型。（仅标准版和企业版支持） • MRS_KAFKA_FORWARDING：转发MRS_KAFKA消息类型。（仅企业版支持） • DMS_ROCKETMQ_FORWARDING：转发RocketMQ消息类型。（仅标准版和企业版支持）

参数	参数类型	描述
channel_detail	**ChannelDetail** object	通道配置信息。

表 1-223 ChannelDetail

参数	参数类型	描述
http_forwarding	**HttpForwarding** object	**参数说明**：http服务器转发消息内容。当channel为HTTP_FORWARDING时，必填。
dis_forwarding	**DisForwarding** object	**参数说明**：转发DIS服务消息内容。当channel为DIS_FORWARDING时，必填。
obs_forwarding	**ObsForwarding** object	**参数说明**：转发OBS服务消息内容。当channel为OBS_FORWARDING时，必填。
amqp_forwarding	**AmqpForwarding** object	**参数说明**：转发AMQP服务消息内容。当channel为AMQP_FORWARDING时，必填。
dms_kafka_forwarding	**DmsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为DMS_KAFKA_FORWARDING时，必填。
roma_forwarding	**RomaForwarding** object	**参数说明**：转发Roma消息内容。当channel为ROMA_FORWARDING时,必填。（仅企业版支持）
mysql_forwarding	**MysqlForwarding** object	**参数说明**：转发MySQL消息内容。当channel为MYSQL_FORWARDING时,必填。
influxdb_forwarding	**InfluxDBForwarding** object	**参数说明**：转发influxdb的配置参数。当channel为INFLUXDB_FORWARDING时，必填。（仅标准版和企业版支持）
functiongraph_forwarding	**FunctionGraphForwarding** object	**参数说明**：转发云服务函数服务消息内容。当channel为FUNCTIONGRAPH_FORWARDING时，必填。（仅标准版和企业版支持）
mrs_kafka_forwarding	**MrsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为MRS_KAFKA_FORWARDING时，必填。（仅企业版支持）
dms_rocketmq_forwarding	**DmsRocketMQForwarding** object	**参数说明**：转发rocketmq消息内容。当channel为DMS_ROCKETMQ_FORWARDING时，必填。（仅标准版和企业版支持）

表 1-224 HttpForwarding

参数	参数类型	描述
url	String	**参数说明**：用于接收满足规则条件数据的http服务器地址。HTTP为非数据加密传输模式，此模式下数据传输不安全， 建议使用更安全的HTTPS方式 最小长度：**1** 最大长度：**256**
cert_id	String	**参数说明**：证书id，请参见**加载推送证书第3步**获取证书ID 最小长度：**1** 最大长度：**64**
cn_name	String	**参数说明**：当sni_enable为true时，此字段需要填写，内容为将要请求的服务端证书的域名,举例:domain:8443;当sni_enbale为false时，此字段默认不填写。 最小长度：**1** 最大长度：**64**
sni_enable	Boolean	**参数说明**：需要https服务端和客户端都支持此功能，默认为false，设成true表明Https的客户端在发起请求时，需要携带cn_name；https服务端根据cn_name返回对应的证书；设为false可关闭此功能。
signature_enable	Boolean	**参数说明**：是否启用签名。填写token时， 该参数必须为true， token才可以生效，否则token不生效。推荐设置成true，使用token签名验证消息是否来自平台。
token	String	**参数说明**：用作生成签名的Token，客户端可以使用该token按照规则生成签名并与推送消息中携带的签名做对比， 从而验证安全性。**取值范围**:长度不超过32， 不小于3， 只允许字母、数字的组合。请参见**HTTP/HTTPS推送基于Token认证物联网平台** 最小长度：**3** 最大长度：**32**

表 1-225 DisForwarding

参数	参数类型	描述
region_name	String	**参数说明**：DIS服务对应的region区域 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
project_id	String	**参数说明**：DIS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
stream_name	String	**参数说明**：DIS服务对应的通道名称，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**
stream_id	String	**参数说明**：DIS服务对应的通道ID，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

表 1-226 ObsForwarding

参数	参数类型	描述
region_name	String	**参数说明**：OBS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：OBS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
bucket_name	String	**参数说明**：OBS服务对应的桶名称 最小长度：**1** 最大长度：**256**
location	String	**参数说明**：OBS服务对应桶的区域 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
file_path	String	**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(/) **取值范围**: 英文字母(a-zA-Z)、数字(0-9)、下划线(_)、中划线(-)、斜杠(/)和大括号({})，最大字符长度256个字符。其中大括号只能用于对应模板参数。 **模板参数**: • {YYYY} 年 • {MM} 月 • {DD} 日 • {HH} 小时 • {appId} 应用ID • {deviceId} 设备ID 例如:自定义目录结构为{YYYY}/{MM}/{DD}/{HH},则会在转发数据时，根据当前时间往对应的目录结构2021>08>11>09下生成对应的数据。

表 1-227 AmqpForwarding

参数	参数类型	描述
queue_name	String	**参数说明**：用于接收满足规则条件数据的amqp queue。 最小长度：**8** 最大长度：**256**

表 1-228 DmsKafkaForwarding

参数	参数类型	描述
region_name	String	**参数说明**：Kafka服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：Kafka服务对应的projectId信息 最小长度：**1** 最大长度：**256**
addresses	Array of **SupportPrivateLinkNetAddress** objects	**参数说明**：转发kafka消息对应的地址列表

参数	参数类型	描述
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发kafka关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发kafka关联的密码信息。 最小长度：**1** 最大长度：**256**
mechanism	String	**参数说明**：转发kafka关联的SASL认证机制。 **取值范围**： ● PAAS：明文传输，此模式下为非数据加密传输模式，数据传输不安全，建议您使用更安全的数据加密模式。 ● PLAIN：SASL/PLAIN模式。需要填写对应的用户名密码信息。一种简单的用户名密码校验机制，在SASL_PLAINTEXT场景下，不建议使用。 ● SCRAM-SHA-512：SASL/SCRAM-SHA-512模式。需要填写对应的用户名密码信息。采用哈希算法对用户名与密码生成凭证，进行身份校验的安全认证机制，比PLAIN机制安全性更高。
security_protocol	String	**参数说明**：kafka传输安全协议，此字段不填默认为SASL_SSL。当mechanism为PAAS或不填时，该字段不生效。 **取值范围**： ● SASL_SSL：采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT：明文传输，支持账号密码认证，性能更好，建议mechanism使用SCRAM-SHA-512机制。

表 1-229 SupportPrivateLinkNetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**

参数	参数类型	描述
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-230 RomaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发roma消息对应的地址列表
topic	String	**参数说明**：转发roma消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发roma关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发roma关联的密码信息。 最大长度：**256**

表 1-231 MysqlForwarding

参数	参数类型	描述
address	**NetAddress** object	转发roma消息对应的地址列表
db_name	String	**参数说明**：连接MYSQL数据库的库名。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合。
username	String	**参数说明**：连接MYSQL数据库的用户名 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：连接MYSQL数据库的密码 最小长度：**1** 最大长度：**256**
enable_ssl	Boolean	**参数说明**：客户端是否使用SSL连接服务端，默认为true， 若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您使用SSL模式。 缺省值：**true**

参数	参数类型	描述
table_name	String	**参数说明**：MYSQL数据库的表名 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：MYSQL数据库的列和流转数据的对应关系列表，最多支持32条映射关系。

表 1-232 InfluxDBForwarding

参数	参数类型	描述
address	**NetAddress** object	**参数说明**：转发InfluxDB消息对应的地址
db_name	String	**参数说明**：连接InfluxDB数据库的库名,不存在会自动创建 最小长度：**1** 最大长度：**64**
username	String	**参数说明**：连接InfluxDB数据库的用户名 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：连接InfluxDB数据库的密码 最小长度：**1** 最大长度：**256**
measurement	String	**参数说明**：InfluxDB数据库的measurement,不存在会自动创建 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：InfluxDB数据库和流转数据的对应关系列表，最多支持32条映射关系。

表 1-233 NetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**

参数	参数类型	描述
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-234 ColumnMapping

参数	参数类型	描述
column_name	String	**参数说明**：数据库的列名 最小长度：**1** 最大长度：**256**
json_key	String	**参数说明**：流转数据的属性名 最小长度：**1** 最大长度：**256**

表 1-235 FunctionGraphForwarding

参数	参数类型	描述
func_urn	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数。 最小长度：**0** 最大长度：**65535**
func_name	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**

表 1-236 MrsKafkaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
kerberos_authentication	Boolean	是否Kerberos认证，默认为false。 缺省值：**false**

表 1-237 DmsRocketMQForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发rocketMQ消息对应的地址列表
topic	String	**参数说明**：转发rocketMQ消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发rocketMQ关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发rocketMQ关联的密码信息。 最小长度：**1** 最大长度：**256**
enable_ssl	Boolean	是否开启SSL，默认为true。若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您开启SSL。 缺省值：**true**

请求示例

列表查询规则动作。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/actions
```

响应示例

状态码： 200

Successful response

```
{
  "actions" : [ {
    "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce1",
    "action_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
    "channel_detail" : {
      "amqp_forwarding" : {
        "queue_name" : "test"
      },
      "obs_forwarding" : {
        "file_path" : "device_property_report/{YYYY}/{MM}/{DD}/{HH}",
        "project_id" : "project_id",
        "bucket_name" : "bucket_name",
        "region_name" : "region_name",
        "location" : "location"
      },
```

```
    "http_forwarding" : {
      "sni_enable" : false,
      "cn_name" : "domain:8443",
      "cert_id" : "0ae892cfeff641158920300b2292d2ca",
      "url" : "http://host:port/callbackurltest"
    },
    "dis_forwarding" : {
      "stream_name" : "stream_name",
      "project_id" : "project_id",
      "stream_id" : "stream_id",
      "region_name" : "region_name"
    },
    "dms_kafka_forwarding" : {
      "addresses" : [ {
        "port" : 443,
        "ip" : "host",
        "domain" : "huawei.com"
      } ],
      "password" : "password",
      "project_id" : "project_id",
      "topic" : "topic",
      "region_name" : "region_name",
      "mechanism" : "PLAIN",
      "security_protocol" : "SASL_SSL",
      "username" : "username"
    }
  },
  "channel" : "HTTP_FORWARDING",
  "app_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce2"
 } ],
 "count" : 10,
 "marker" : "5c90fa7d3c4e4405e8525079"
}
```


SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListRuleActionsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
```

```
            .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
            .withAk(ak)
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        ListRuleActionsRequest request = new ListRuleActionsRequest();
        request.withRuleId("<rule_id>");
        request.withChannel("<channel>");
        request.withAppType("<app_type>");
        request.withAppId("<app_id>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListRuleActionsResponse response = client.listRuleActions(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
```



```
        request = ListRuleActionsRequest()
        request.rule_id = "<rule_id>"
        request.channel = "<channel>"
        request.app_type = "<app_type>"
        request.app_id = "<app_id>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_rule_actions(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListRuleActionsRequest{}
    ruleIdRequest:= "<rule_id>"
    request.RuleId = &ruleIdRequest
    channelRequest:= "<channel>"
    request.Channel = &channelRequest
    appTypeRequest:= "<app_type>"
    request.AppType = &appTypeRequest
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
```



```
    request.Offset = &offsetRequest
    response, err := client.ListRuleActions(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Successful response
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.8 查询规则动作

功能介绍

应用服务器可调用此接口查询物联网平台中指定规则动作的配置信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/actions/{action_id}

表 1-238 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

参数	是否必选	参数类型	描述
action_id	是	String	**参数说明**：规则动作ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-239 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-240 响应 Body 参数

参数	参数类型	描述
action_id	String	规则动作ID，用于唯一标识一条规则动作，在创建规则动作时由物联网平台分配获得，创建时无需携带，由平台统一分配唯一的action_id。
rule_id	String	规则动作对应的的规则触发条件ID。
app_id	String	资源空间ID。

参数	参数类型	描述
channel	String	规则动作的类型，取值范围： • HTTP_FORWARDING：HTTP服务消息类型。 • DIS_FORWARDING：转发DIS服务消息类型。 • OBS_FORWARDING：转发OBS服务消息类型。 • AMQP_FORWARDING：转发AMQP服务消息类型。 • DMS_KAFKA_FORWARDING：转发kafka消息类型。 • ROMA_FORWARDING: 转发Roma消息类型。（仅企业版支持） • INFLUXDB_FORWARDING：转发InfluxDB消息类型。（仅标准版和企业版支持） • MYSQL_FORWARDING：转发MySQL消息类型。（仅标准版和企业版支持） • FUNCTIONGRAPH_FORWARDING：转发FunctionGraph消息类型。（仅标准版和企业版支持） • MRS_KAFKA_FORWARDING：转发MRS_KAFKA消息类型。（仅企业版支持） • DMS_ROCKETMQ_FORWARDING：转发RocketMQ消息类型。（仅标准版和企业版支持）
channel_detail	**ChannelDetail** object	通道配置信息。

表 1-241 ChannelDetail

参数	参数类型	描述
http_forwarding	**HttpForwarding** object	**参数说明**：http服务器转发消息内容。当channel为HTTP_FORWARDING时，必填。
dis_forwarding	**DisForwarding** object	**参数说明**：转发DIS服务消息内容。当channel为DIS_FORWARDING时，必填。
obs_forwarding	**ObsForwarding** object	**参数说明**：转发OBS服务消息内容。当channel为OBS_FORWARDING时，必填。
amqp_forwarding	**AmqpForwarding** object	**参数说明**：转发AMQP服务消息内容。当channel为AMQP_FORWARDING时，必填。
dms_kafka_forwarding	**DmsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为DMS_KAFKA_FORWARDING时，必填。

参数	参数类型	描述
roma_forwarding	**RomaForwarding** object	**参数说明**：转发Roma消息内容。当channel为ROMA_FORWARDING时,必填。（仅企业版支持）
mysql_forwarding	**MysqlForwarding** object	**参数说明**：转发MySQL消息内容。当channel为MYSQL_FORWARDING时,必填。
influxdb_forwarding	**InfluxDBForwarding** object	**参数说明**：转发influxdb的配置参数。当channel为INFLUXDB_FORWARDING时，必填。（仅标准版和企业版支持）
functiongraph_forwarding	**FunctionGraphForwarding** object	**参数说明**：转发云服务函数服务消息内容。当channel为FUNCTIONGRAPH_FORWARDING时，必填。（仅标准版和企业版支持）
mrs_kafka_forwarding	**MrsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为MRS_KAFKA_FORWARDING时，必填。（仅企业版支持）
dms_rocketmq_forwarding	**DmsRocketMQForwarding** object	**参数说明**：转发rocketmq消息内容。当channel为DMS_ROCKETMQ_FORWARDING时，必填。（仅标准版和企业版支持）

表 1-242 HttpForwarding

参数	参数类型	描述
url	String	**参数说明**：用于接收满足规则条件数据的http服务器地址。HTTP为非数据加密传输模式，此模式下数据传输不安全， 建议使用更安全的HTTPS方式 最小长度：**1** 最大长度：**256**
cert_id	String	**参数说明**：证书id，请参见**加载推送证书第3步**获取证书ID 最小长度：**1** 最大长度：**64**
cn_name	String	**参数说明**：当sni_enable为true时，此字段需要填写，内容为将要请求的服务端证书的域名,举例:domain:8443;当sni_enbale为false时，此字段默认不填写。 最小长度：**1** 最大长度：**64**

参数	参数类型	描述
sni_enable	Boolean	**参数说明**：需要https服务端和客户端都支持此功能，默认为false，设成true表明Https的客户端在发起请求时，需要携带cn_name；https服务端根据cn_name返回对应的证书；设为false可关闭此功能。
signature_enable	Boolean	**参数说明**：是否启用签名。填写token时， 该参数必须为true， token才可以生效，否则token不生效。推荐设置成true，使用token签名验证消息是否来自平台。
token	String	**参数说明**：用作生成签名的Token，客户端可以使用该token按照规则生成签名并与推送消息中携带的签名做对比， 从而验证安全性。**取值范围**: 长度不超过32， 不小于3， 只允许字母、数字的组合。请参见**HTTP/HTTPS推送基于Token认证 物联网平台** 最小长度：**3** 最大长度：**32**

表 1-243 DisForwarding

参数	参数类型	描述
region_name	String	**参数说明**：DIS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：DIS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
stream_name	String	**参数说明**：DIS服务对应的通道名称，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**
stream_id	String	**参数说明**：DIS服务对应的通道ID，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

表 1-244 ObsForwarding

参数	参数类型	描述
region_name	String	**参数说明**：OBS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：OBS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
bucket_name	String	**参数说明**：OBS服务对应的桶名称 最小长度：**1** 最大长度：**256**
location	String	**参数说明**：OBS服务对应桶的区域 最小长度：**1** 最大长度：**256**
file_path	String	**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(/) **取值范围**: 英文字母(a-zA-Z)、数字(0-9)、下划线(_)、中划线(-)、斜杠(/)和大括号({})，最大字符长度256个字符。其中大括号只能用于对应模板参数。 **模板参数**: ● {YYYY} 年 ● {MM} 月 ● {DD} 日 ● {HH} 小时 ● {appId} 应用ID ● {deviceId} 设备ID 例如:自定义目录结构为{YYYY}/{MM}/{DD}/{HH},则会在转发数据时，根据当前时间往对应的目录结构2021>08>11>09下生成对应的数据。

表 1-245 AmqpForwarding

参数	参数类型	描述
queue_name	String	**参数说明**：用于接收满足规则条件数据的amqp queue。 最小长度：**8** 最大长度：**256**

表 1-246 DmsKafkaForwarding

参数	参数类型	描述
region_name	String	**参数说明**：Kafka服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：Kafka服务对应的projectId信息 最小长度：**1** 最大长度：**256**
addresses	Array of **SupportPrivateLinkNetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发kafka关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发kafka关联的密码信息。 最小长度：**1** 最大长度：**256**
mechanism	String	**参数说明**：转发kafka关联的SASL认证机制。 **取值范围**： • PAAS：明文传输，此模式下为非数据加密传输模式，数据传输不安全，建议您使用更安全的数据加密模式。 • PLAIN：SASL/PLAIN模式。需要填写对应的用户名密码信息。一种简单的用户名密码校验机制，在SASL_PLAINTEXT场景下，不建议使用。 • SCRAM-SHA-512：SASL/SCRAM-SHA-512模式。需要填写对应的用户名密码信息。采用哈希算法对用户名与密码生成凭证，进行身份校验的安全认证机制，比PLAIN机制安全性更高。

参数	参数类型	描述
security_protocol	String	**参数说明**：kafka传输安全协议，此字段不填默认为SASL_SSL。当mechanism为PAAS或不填时，该字段不生效。 **取值范围**： • SASL_SSL：采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 • SASL_PLAINTEXT：明文传输，支持账号密码认证，性能更好，建议mechanism使用SCRAM-SHA-512机制。

表 1-247 SupportPrivateLinkNetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-248 RomaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发roma消息对应的地址列表
topic	String	**参数说明**：转发roma消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发roma关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发roma关联的密码信息。 最大长度：**256**

表 1-249 MysqlForwarding

参数	参数类型	描述
address	**NetAddress** object	转发roma消息对应的地址列表
db_name	String	**参数说明**：连接MYSQL数据库的库名。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合。
username	String	**参数说明**：连接MYSQL数据库的用户名 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：连接MYSQL数据库的密码 最小长度：**1** 最大长度：**256**
enable_ssl	Boolean	**参数说明**：客户端是否使用SSL连接服务端，默认为true， 若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您使用SSL模式。 缺省值：**true**
table_name	String	**参数说明**：MYSQL数据库的表名 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：MYSQL数据库的列和流转数据的对应关系列表，最多支持32条映射关系。

表 1-250 InfluxDBForwarding

参数	参数类型	描述
address	**NetAddress** object	**参数说明**：转发InfluxDB消息对应的地址
db_name	String	**参数说明**：连接InfluxDB数据库的库名,不存在会自动创建 最小长度：**1** 最大长度：**64**
username	String	**参数说明**：连接InfluxDB数据库的用户名 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
password	String	**参数说明**：连接InfluxDB数据库的密码 最小长度：**1** 最大长度：**256**
measurement	String	**参数说明**：InfluxDB数据库的measurement,不存在会自动创建 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：InfluxDB数据库和流转数据的对应关系列表，最多支持32条映射关系。

表 1-251 NetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-252 ColumnMapping

参数	参数类型	描述
column_name	String	**参数说明**：数据库的列名 最小长度：**1** 最大长度：**256**
json_key	String	**参数说明**：流转数据的属性名 最小长度：**1** 最大长度：**256**

表 1-253 FunctionGraphForwarding

参数	参数类型	描述
func_urn	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数。 最小长度：**0** 最大长度：**65535**
func_name	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**

表 1-254 MrsKafkaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
kerberos_authentication	Boolean	是否Kerberos认证，默认为false。 缺省值：**false**

表 1-255 DmsRocketMQForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发rocketMQ消息对应的地址列表
topic	String	**参数说明**：转发rocketMQ消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发rocketMQ关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发rocketMQ关联的密码信息。 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
enable_ssl	Boolean	是否开启SSL，默认为true。若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您开启SSL。 缺省值：**true**

请求示例

查询指定规则动作详情。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/actions/{action_id}
```

响应示例

状态码： 200

OK

```
{
  "action_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce1",
  "app_id" : "1a7ffc5cd89c44dd8265b1653d951ce0",
  "channel" : "HTTP_FORWARDING",
  "channel_detail" : {
    "http_forwarding" : {
      "url" : "http://host:port/callbackurltest"
    }
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowRuleActionSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
```



```
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowRuleActionRequest request = new ShowRuleActionRequest();
        try {
            ShowRuleActionResponse response = client.showRuleAction(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowRuleActionRequest()
        response = client.show_rule_action(request)
        print(response)
```



```
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowRuleActionRequest{}
    response, err := client.ShowRuleAction(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK

状态码	描述
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.9 修改规则动作

功能介绍

应用服务器可调用此接口修改物联网平台中指定规则动作的配置。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/routing-rule/actions/{action_id}

表 1-256 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
action_id	是	String	**参数说明**：规则动作ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-257 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-258 请求 Body 参数

参数	是否必选	参数类型	描述
channel	否	String	**参数说明**：规则动作的类型。 **取值范围**： • HTTP_FORWARDING：HTTP服务消息类型。 • DIS_FORWARDING：转发DIS服务消息类型。 • OBS_FORWARDING：转发OBS服务消息类型。 • AMQP_FORWARDING：转发AMQP服务消息类型。 • DMS_KAFKA_FORWARDING：转发kafka消息类型。 • ROMA_FORWARDING：转发Roma消息类型。（仅企业版支持） • INFLUXDB_FORWARDING：转发InfluxDB消息类型。（仅标准版和企业版支持） • MYSQL_FORWARDING：转发MySQL消息类型。（仅标准版和企业版支持） • FUNCTIONGRAPH_FORWARDING：转发FunctionGraph消息类型。（仅标准版和企业版支持） • MRS_KAFKA_FORWARDING：转发MRS_KAFKA消息类型。（仅企业版支持） • DMS_ROCKETMQ_FORWARDING：转发RocketMQ消息类型。（仅标准版和企业版支持）
channel_detail	否	ChannelDetail object	**参数说明**：通道配置信息。

表 1-259 ChannelDetail

参数	是否必选	参数类型	描述
http_forwarding	否	HttpForwarding object	**参数说明**：http服务器转发消息内容。当channel为HTTP_FORWARDING时，必填。

参数	是否必选	参数类型	描述
dis_forwarding	否	**DisForwarding** object	**参数说明**：转发DIS服务消息内容。当channel为DIS_FORWARDING时，必填。
obs_forwarding	否	**ObsForwarding** object	**参数说明**：转发OBS服务消息内容。当channel为OBS_FORWARDING时，必填。
amqp_forwarding	否	**AmqpForwarding** object	**参数说明**：转发AMQP服务消息内容。当channel为AMQP_FORWARDING时，必填。
dms_kafka_forwarding	否	**DmsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为DMS_KAFKA_FORWARDING时，必填。
roma_forwarding	否	**RomaForwarding** object	**参数说明**：转发Roma消息内容。当channel为ROMA_FORWARDING时,必填。（仅企业版支持）
mysql_forwarding	否	**MysqlForwarding** object	**参数说明**：转发MySQL消息内容。当channel为MYSQL_FORWARDING时,必填。
influxdb_forwarding	否	**InfluxDBForwarding** object	**参数说明**：转发influxdb的配置参数。当channel为INFLUXDB_FORWARDING时，必填。（仅标准版和企业版支持）
functiongraph_forwarding	否	**FunctionGraphForwarding** object	**参数说明**：转发云服务函数服务消息内容。当channel为FUNCTIONGRAPH_FORWARDING时，必填。（仅标准版和企业版支持）
mrs_kafka_forwarding	否	**MrsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为MRS_KAFKA_FORWARDING时，必填。（仅企业版支持）
dms_rocketmq_forwarding	否	**DmsRocketMQForwarding** object	**参数说明**：转发rocketmq消息内容。当channel为DMS_ROCKETMQ_FORWARDING时，必填。（仅标准版和企业版支持）

表 1-260 HttpForwarding

参数	是否必选	参数类型	描述
url	是	String	**参数说明**：用于接收满足规则条件数据的http服务器地址。HTTP为非数据加密传输模式，此模式下数据传输不安全， 建议使用更安全的HTTPS方式 最小长度：**1** 最大长度：**256**
cert_id	否	String	**参数说明**：证书id，请参见**加载推送证书第3步**获取证书ID 最小长度：**1** 最大长度：**64**
cn_name	否	String	**参数说明**：当sni_enable为true时，此字段需要填写，内容为将要请求的服务端证书的域名,举例:domain:8443;当sni_enbale为false时，此字段默认不填写。 最小长度：**1** 最大长度：**64**
sni_enable	否	Boolean	**参数说明**：需要https服务端和客户端都支持此功能，默认为false，设成true表明Https的客户端在发起请求时，需要携带cn_name；https服务端根据cn_name返回对应的证书；设为false可关闭此功能。
signature_enable	否	Boolean	**参数说明**：是否启用签名。填写token时， 该参数必须为true，token才可以生效，否则token不生效。推荐设置成true，使用token签名验证消息是否来自平台。
token	否	String	**参数说明**：用作生成签名的Token，客户端可以使用该token按照规则生成签名并与推送消息中携带的签名做对比，从而验证安全性。**取值范围**: 长度不超过32， 不小于3， 只允许字母、数字的组合。请参见**HTTP/HTTPS推送基于Token认证物联网平台** 最小长度：**3** 最大长度：**32**

表 1-261 DisForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：DIS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	是	String	**参数说明**：DIS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
stream_name	否	String	**参数说明**：DIS服务对应的通道名称，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**
stream_id	否	String	**参数说明**：DIS服务对应的通道ID，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

表 1-262 ObsForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：OBS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	是	String	**参数说明**：OBS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
bucket_name	是	String	**参数说明**：OBS服务对应的桶名称 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
location	否	String	**参数说明**：OBS服务对应桶的区域 最小长度：**1** 最大长度：**256**
file_path	否	String	**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(/) **取值范围**: 英文字母(a-zA-Z)、数字(0-9)、下划线(_)、中划线(-)、斜杠(/)和大括号({})，最大字符长度256个字符。其中大括号只能用于对应模板参数。 **模板参数**: ● {YYYY} 年 ● {MM} 月 ● {DD} 日 ● {HH} 小时 ● {appId} 应用ID ● {deviceId} 设备ID 例如:自定义目录结构为{YYYY}/{MM}/{DD}/{HH},则会在转发数据时，根据当前时间往对应的目录结构2021>08>11>09下生成对应的数据。

表 1-263 AmqpForwarding

参数	是否必选	参数类型	描述
queue_name	是	String	**参数说明**：用于接收满足规则条件数据的amqp queue。 最小长度：**8** 最大长度：**256**

表 1-264 DmsKafkaForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：Kafka服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	是	String	**参数说明**：Kafka服务对应的projectId信息 最小长度：**1** 最大长度：**256**
addresses	是	Array of **SupportPrivateLinkNetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	是	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	否	String	**参数说明**：转发kafka关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	否	String	**参数说明**：转发kafka关联的密码信息。 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
mechanism	否	String	**参数说明**：转发kafka关联的SASL认证机制。 **取值范围**： ● PAAS：明文传输，此模式下为非数据加密传输模式，数据传输不安全，建议您使用更安全的数据加密模式。 ● PLAIN：SASL/PLAIN模式。需要填写对应的用户名密码信息。一种简单的用户名密码校验机制，在SASL_PLAINTEXT场景下，不建议使用。 ● SCRAM-SHA-512：SASL/SCRAM-SHA-512模式。需要填写对应的用户名密码信息。采用哈希算法对用户名与密码生成凭证，进行身份校验的安全认证机制，比PLAIN机制安全性更高。
security_protocol	否	String	**参数说明**：kafka传输安全协议，此字段不填默认为SASL_SSL。当mechanism为PAAS或不填时，该字段不生效。 **取值范围**： ● SASL_SSL：采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT：明文传输，支持账号密码认证，性能更好，建议mechanism使用SCRAM-SHA-512机制。

表 1-265 SupportPrivateLinkNetAddress

参数	是否必选	参数类型	描述
ip	否	String	**参数说明**：服务的对应IP
port	否	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	否	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-266 RomaForwarding

参数	是否必选	参数类型	描述
addresses	是	Array of **NetAddress** objects	**参数说明**：转发roma消息对应的地址列表
topic	是	String	**参数说明**：转发roma消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	是	String	**参数说明**：转发roma关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：转发roma关联的密码信息。 最大长度：**256**

表 1-267 MysqlForwarding

参数	是否必选	参数类型	描述
address	是	**NetAddress** object	转发roma消息对应的地址列表
db_name	是	String	**参数说明**：连接MYSQL数据库的库名。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合。
username	是	String	**参数说明**：连接MYSQL数据库的用户名 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：连接MYSQL数据库的密码 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
enable_ssl	否	Boolean	**参数说明**：客户端是否使用SSL连接服务端，默认为true， 若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您使用SSL模式。 缺省值：**true**
table_name	是	String	**参数说明**：MYSQL数据库的表名 最小长度：**1** 最大长度：**64**
column_mappings	是	Array of **ColumnMapping** objects	**参数说明**：MYSQL数据库的列和流转数据的对应关系列表，最多支持32条映射关系。

表 1-268 InfluxDBForwarding

参数	是否必选	参数类型	描述
address	是	**NetAddress** object	**参数说明**：转发InfluxDB消息对应的地址
db_name	是	String	**参数说明**：连接InfluxDB数据库的库名,不存在会自动创建 最小长度：**1** 最大长度：**64**
username	是	String	**参数说明**：连接InfluxDB数据库的用户名 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：连接InfluxDB数据库的密码 最小长度：**1** 最大长度：**256**
measurement	是	String	**参数说明**：InfluxDB数据库的measurement,不存在会自动创建 最小长度：**1** 最大长度：**64**
column_mappings	是	Array of **ColumnMapping** objects	**参数说明**：InfluxDB数据库和流转数据的对应关系列表，最多支持32条映射关系。

表 1-269 NetAddress

参数	是否必选	参数类型	描述
ip	否	String	**参数说明**：服务的对应IP
port	否	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	否	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-270 ColumnMapping

参数	是否必选	参数类型	描述
column_name	是	String	**参数说明**：数据库的列名 最小长度：**1** 最大长度：**256**
json_key	是	String	**参数说明**：流转数据的属性名 最小长度：**1** 最大长度：**256**

表 1-271 FunctionGraphForwarding

参数	是否必选	参数类型	描述
func_urn	是	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数。 最小长度：**0** 最大长度：**65535**
func_name	是	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**

表 1-272 MrsKafkaForwarding

参数	是否必选	参数类型	描述
addresses	是	Array of **NetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	是	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
kerberos_authentication	否	Boolean	是否Kerberos认证，默认为false。 缺省值：**false**

表 1-273 DmsRocketMQForwarding

参数	是否必选	参数类型	描述
addresses	是	Array of **NetAddress** objects	**参数说明**：转发rocketMQ消息对应的地址列表
topic	是	String	**参数说明**：转发rocketMQ消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	是	String	**参数说明**：转发rocketMQ关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	是	String	**参数说明**：转发rocketMQ关联的密码信息。 最小长度：**1** 最大长度：**256**
enable_ssl	否	Boolean	是否开启SSL，默认为true。若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您开启SSL。 缺省值：**true**

响应参数

状态码： 200

表 1-274 响应 Body 参数

参数	参数类型	描述
action_id	String	规则动作ID，用于唯一标识一条规则动作，在创建规则动作时由物联网平台分配获得，创建时无需携带，由平台统一分配唯一的action_id。
rule_id	String	规则动作对应的的规则触发条件ID。
app_id	String	资源空间ID。
channel	String	规则动作的类型，取值范围： • HTTP_FORWARDING：HTTP服务消息类型。 • DIS_FORWARDING：转发DIS服务消息类型。 • OBS_FORWARDING：转发OBS服务消息类型。 • AMQP_FORWARDING：转发AMQP服务消息类型。 • DMS_KAFKA_FORWARDING：转发kafka消息类型。 • ROMA_FORWARDING: 转发Roma消息类型。（仅企业版支持） • INFLUXDB_FORWARDING：转发InfluxDB消息类型。（仅标准版和企业版支持） • MYSQL_FORWARDING：转发MySQL消息类型。（仅标准版和企业版支持） • FUNCTIONGRAPH_FORWARDING：转发FunctionGraph消息类型。（仅标准版和企业版支持） • MRS_KAFKA_FORWARDING：转发MRS_KAFKA消息类型。（仅企业版支持） • DMS_ROCKETMQ_FORWARDING：转发RocketMQ消息类型。（仅标准版和企业版支持）
channel_detail	**ChannelDetail** object	通道配置信息。

表 1-275 ChannelDetail

参数	参数类型	描述
http_forwarding	**HttpForwarding** object	**参数说明**：http服务器转发消息内容。当channel为HTTP_FORWARDING时，必填。
dis_forwarding	**DisForwarding** object	**参数说明**：转发DIS服务消息内容。当channel为DIS_FORWARDING时，必填。

参数	参数类型	描述
obs_forwarding	**ObsForwarding** object	**参数说明**：转发OBS服务消息内容。当channel为OBS_FORWARDING时，必填。
amqp_forwarding	**AmqpForwarding** object	**参数说明**：转发AMQP服务消息内容。当channel为AMQP_FORWARDING时，必填。
dms_kafka_forwarding	**DmsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为DMS_KAFKA_FORWARDING时，必填。
roma_forwarding	**RomaForwarding** object	**参数说明**：转发Roma消息内容。当channel为ROMA_FORWARDING时,必填。（仅企业版支持）
mysql_forwarding	**MysqlForwarding** object	**参数说明**：转发MySQL消息内容。当channel为MYSQL_FORWARDING时,必填。
influxdb_forwarding	**InfluxDBForwarding** object	**参数说明**：转发influxdb的配置参数。当channel为INFLUXDB_FORWARDING时，必填。（仅标准版和企业版支持）
functiongraph_forwarding	**FunctionGraphForwarding** object	**参数说明**：转发云服务函数服务消息内容。当channel为FUNCTIONGRAPH_FORWARDING时，必填。（仅标准版和企业版支持）
mrs_kafka_forwarding	**MrsKafkaForwarding** object	**参数说明**：转发Kafka消息内容。当channel为MRS_KAFKA_FORWARDING时，必填。（仅企业版支持）
dms_rocketmq_forwarding	**DmsRocketMQForwarding** object	**参数说明**：转发rocketmq消息内容。当channel为DMS_ROCKETMQ_FORWARDING时，必填。（仅标准版和企业版支持）

表 1-276 HttpForwarding

参数	参数类型	描述
url	String	**参数说明**：用于接收满足规则条件数据的http服务器地址。HTTP为非数据加密传输模式，此模式下数据传输不安全， 建议使用更安全的HTTPS方式 最小长度：**1** 最大长度：**256**
cert_id	String	**参数说明**：证书id，请参见**加载推送证书第3步**获取证书ID 最小长度：**1** 最大长度：**64**

参数	参数类型	描述
cn_name	String	**参数说明**：当sni_enable为true时，此字段需要填写，内容为将要请求的服务端证书的域名,举例:domain:8443;当sni_enbale为false时，此字段默认不填写。 最小长度：**1** 最大长度：**64**
sni_enable	Boolean	**参数说明**：需要https服务端和客户端都支持此功能，默认为false，设成true表明Https的客户端在发起请求时，需要携带cn_name；https服务端根据cn_name返回对应的证书；设为false可关闭此功能。
signature_enable	Boolean	**参数说明**：是否启用签名。填写token时， 该参数必须为true， token才可以生效，否则token不生效。推荐设置成true，使用token签名验证消息是否来自平台。
token	String	**参数说明**：用作生成签名的Token，客户端可以使用该token按照规则生成签名并与推送消息中携带的签名做对比， 从而验证安全性。**取值范围**:长度不超过32， 不小于3， 只允许字母、数字的组合。请参见**HTTP/HTTPS推送基于Token认证 物联网平台** 最小长度：**3** 最大长度：**32**

表 1-277 DisForwarding

参数	参数类型	描述
region_name	String	**参数说明**：DIS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：DIS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
stream_name	String	**参数说明**：DIS服务对应的通道名称，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
stream_id	String	**参数说明**：DIS服务对应的通道ID，stream_id和stream_name两个参数必须携带一个，优先使用stream_id 最小长度：**1** 最大长度：**256**

表 1-278 ObsForwarding

参数	参数类型	描述
region_name	String	**参数说明**：OBS服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：OBS服务对应的projectId信息 最小长度：**1** 最大长度：**256**
bucket_name	String	**参数说明**：OBS服务对应的桶名称 最小长度：**1** 最大长度：**256**
location	String	**参数说明**：OBS服务对应桶的区域 最小长度：**1** 最大长度：**256**
file_path	String	**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(/) **取值范围**: 英文字母(a-zA-Z)、数字(0-9)、下划线(_)、中划线(-)、斜杠(/)和大括号({})，最大字符长度256个字符。其中大括号只能用于对应模板参数。 **模板参数**: ● {YYYY} 年 ● {MM} 月 ● {DD} 日 ● {HH} 小时 ● {appId} 应用ID ● {deviceId} 设备ID 例如:自定义目录结构为{YYYY}/{MM}/{DD}/{HH},则会在转发数据时，根据当前时间往对应的目录结构2021>08>11>09下生成对应的数据。

表 1-279 AmqpForwarding

参数	参数类型	描述
queue_name	String	**参数说明**：用于接收满足规则条件数据的amqp queue。 最小长度：**8** 最大长度：**256**

表 1-280 DmsKafkaForwarding

参数	参数类型	描述
region_name	String	**参数说明**：Kafka服务对应的region区域 最小长度：**1** 最大长度：**256**
project_id	String	**参数说明**：Kafka服务对应的projectId信息 最小长度：**1** 最大长度：**256**
addresses	Array of **SupportPrivateLinkNetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发kafka关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发kafka关联的密码信息。 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
mechanism	String	**参数说明**：转发kafka关联的SASL认证机制。 **取值范围**： ● PAAS：明文传输，此模式下为非数据加密传输模式，数据传输不安全，建议您使用更安全的数据加密模式。 ● PLAIN：SASL/PLAIN模式。需要填写对应的用户名密码信息。一种简单的用户名密码校验机制，在SASL_PLAINTEXT场景下，不建议使用。 ● SCRAM-SHA-512：SASL/SCRAM-SHA-512模式。需要填写对应的用户名密码信息。采用哈希算法对用户名与密码生成凭证，进行身份校验的安全认证机制，比PLAIN机制安全性更高。
security_protocol	String	**参数说明**：kafka传输安全协议，此字段不填默认为SASL_SSL。当mechanism为PAAS或不填时，该字段不生效。 **取值范围**： ● SASL_SSL：采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT：明文传输，支持账号密码认证，性能更好，建议mechanism使用SCRAM-SHA-512机制。

表 1-281 SupportPrivateLinkNetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-282 RomaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发roma消息对应的地址列表

参数	参数类型	描述
topic	String	**参数说明**：转发roma消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发roma关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发roma关联的密码信息。 最大长度：**256**

表 1-283 MysqlForwarding

参数	参数类型	描述
address	**NetAddress** object	转发roma消息对应的地址列表
db_name	String	**参数说明**：连接MYSQL数据库的库名。 **取值范围**：长度不超过64，只允许字母、数字、下划线（_）、连接符（-）的组合。
username	String	**参数说明**：连接MYSQL数据库的用户名 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：连接MYSQL数据库的密码 最小长度：**1** 最大长度：**256**
enable_ssl	Boolean	**参数说明**：客户端是否使用SSL连接服务端，默认为true， 若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您使用SSL模式。 缺省值：**true**
table_name	String	**参数说明**：MYSQL数据库的表名 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：MYSQL数据库的列和流转数据的对应关系列表，最多支持32条映射关系。

表 1-284 InfluxDBForwarding

参数	参数类型	描述
address	**NetAddress** object	**参数说明**：转发InfluxDB消息对应的地址
db_name	String	**参数说明**：连接InfluxDB数据库的库名,不存在会自动创建 最小长度：**1** 最大长度：**64**
username	String	**参数说明**：连接InfluxDB数据库的用户名 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：连接InfluxDB数据库的密码 最小长度：**1** 最大长度：**256**
measurement	String	**参数说明**：InfluxDB数据库的measurement,不存在会自动创建 最小长度：**1** 最大长度：**64**
column_mappings	Array of **ColumnMapping** objects	**参数说明**：InfluxDB数据库和流转数据的对应关系列表，最多支持32条映射关系。

表 1-285 NetAddress

参数	参数类型	描述
ip	String	**参数说明**：服务的对应IP
port	Integer	**参数说明**：服务对应端口 最小值：**0** 最大值：**65535**
domain	String	**参数说明**：服务对应的域名 最小长度：**4** 最大长度：**255**

表 1-286 ColumnMapping

参数	参数类型	描述
column_name	String	**参数说明**：数据库的列名 最小长度：**1** 最大长度：**256**
json_key	String	**参数说明**：流转数据的属性名 最小长度：**1** 最大长度：**256**

表 1-287 FunctionGraphForwarding

参数	参数类型	描述
func_urn	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数。 最小长度：**0** 最大长度：**65535**
func_name	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**

表 1-288 MrsKafkaForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发kafka消息对应的地址列表
topic	String	**参数说明**：转发kafka消息关联的topic信息。 最小长度：**1** 最大长度：**256**
kerberos_authentication	Boolean	是否Kerberos认证，默认为false。 缺省值：**false**

表 1-289 DmsRocketMQForwarding

参数	参数类型	描述
addresses	Array of **NetAddress** objects	**参数说明**：转发rocketMQ消息对应的地址列表
topic	String	**参数说明**：转发rocketMQ消息关联的topic信息。 最小长度：**1** 最大长度：**256**
username	String	**参数说明**：转发rocketMQ关联的用户名信息。 最小长度：**1** 最大长度：**256**
password	String	**参数说明**：转发rocketMQ关联的密码信息。 最小长度：**1** 最大长度：**256**
enable_ssl	Boolean	是否开启SSL，默认为true。若为false，则为非数据加密传输模式，此模式下数据传输不安全，建议您开启SSL。 缺省值：**true**

请求示例

修改指定规则动作， 推送到http服务器。

```
PUT https://{endpoint}/v5/iot/{project_id}/routing-rule/actions/{action_id}

{
  "channel" : "HTTP_FORWARDING",
  "channel_detail" : {
    "http_forwarding" : {
      "url" : "http://host:port/callbackurltest"
    }
  }
}
```

响应示例

状态码： 200

OK

```
{
  "action_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "rule_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce1",
  "app_id" : "1a7ffc5cd89c44dd8265b1653d951ce0",
  "channel" : "HTTP_FORWARDING",
  "channel_detail" : {
    "http_forwarding" : {
      "url" : "http://host:port/callbackurltest"
    }
```



```
 }
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改指定规则动作， 推送到http服务器。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateRuleActionSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateRuleActionRequest request = new UpdateRuleActionRequest();
        UpdateActionReq body = new UpdateActionReq();
        HttpForwarding httpForwardingChannelDetail = new HttpForwarding();
        httpForwardingChannelDetail.withUrl("http://host:port/callbackurltest");
        ChannelDetail channelDetailbody = new ChannelDetail();
        channelDetailbody.withHttpForwarding(httpForwardingChannelDetail);
        body.withChannelDetail(channelDetailbody);
        body.withChannel("HTTP_FORWARDING");
        request.withBody(body);
        try {
            UpdateRuleActionResponse response = client.updateRuleAction(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
```



```
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改指定规则动作， 推送到http服务器。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateRuleActionRequest()
        httpForwardingChannelDetail = HttpForwarding(
            url="http://host:port/callbackurltest"
        )
        channelDetailbody = ChannelDetail(
            http_forwarding=httpForwardingChannelDetail
        )
        request.body = UpdateActionReq(
            channel_detail=channelDetailbody,
            channel="HTTP_FORWARDING"
        )
        response = client.update_rule_action(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改指定规则动作， 推送到http服务器。

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateRuleActionRequest{}
    httpForwardingChannelDetail := &model.HttpForwarding{
        Url: "http://host:port/callbackurltest",
    }
    channelDetailbody := &model.ChannelDetail{
        HttpForwarding: httpForwardingChannelDetail,
    }
    channelUpdateActionReq:= "HTTP_FORWARDING"
    request.Body = &model.UpdateActionReq{
        ChannelDetail: channelDetailbody,
        Channel: &channelUpdateActionReq,
    }
    response, err := client.UpdateRuleAction(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK

状态码	描述
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.8.10 删除规则动作

功能介绍

应用服务器可调用此接口删除物联网平台中的指定规则动作。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/routing-rule/actions/{action_id}

表 1-290 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
action_id	是	String	**参数说明**：规则动作ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-291 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

无

请求示例

删除指定规则动作。

```
DELETE https://{endpoint}/v5/iot/{project_id}/routing-rule/actions/{action_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```

```
public class DeleteRuleActionSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteRuleActionRequest request = new DeleteRuleActionRequest();
        try {
            DeleteRuleActionResponse response = client.deleteRuleAction(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteRuleActionRequest()
        response = client.delete_rule_action(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteRuleActionRequest{}
    response, err := client.DeleteRuleAction(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.9 流转数据

1.4.9.1 设备状态变更通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device.status，Event:update）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当物联网平台中的设备状态发生变更时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备状态变更通知规则的转发目标决定

请求参数

表 1-292 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device.status。
event	是	String	**参数说明**：订阅的资源事件，取值为update。

参数	是否必选	参数类型	描述
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DeviceStatusUpdateNotifyDataV5** object	**参数说明**：推送消息。

表 1-293 DeviceStatusUpdateNotifyDataV5

参数	是否必选	参数类型	描述
header	是	**NotifyDataHeader** object	**参数说明**：推送消息header。
body	是	**DeviceStatusUpdate** object	**参数说明**：推送消息body。

表 1-294 NotifyDataHeader

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：应用ID。 最大长度：**256**
device_id	否	String	**参数说明**：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。 最大长度：**256**

参数	是否必选	参数类型	描述
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**256**
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在注册产品时由物联网平台分配获得。 最大长度：**256**
gateway_id	否	String	**参数说明**：网关ID，用于标识一个网关设备。当设备是直连设备时，gatewayId与设备的deviceId一致。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 最大长度：**256**
tags	否	Array of **TagV5DTO** objects	**参数说明**：要绑定到指定资源的标签列表，标签列表中各项标签键值之间不允许重复，一个资源最多可以绑定10个标签。

表 1-295 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

表 1-296 DeviceStatusUpdate

参数	是否必选	参数类型	描述
status	是	String	**参数说明**：设备当前的状态。 • ONLINE：设备在线 • OFFLINE：设备离线 • ABNORMAL：设备异常
last_online_time	是	String	**参数说明**：最近一次上线时间。

参数	是否必选	参数类型	描述
status_update_time	是	String	**参数说明**：设备更新到当前状态的时间。

响应参数

无

请求示例

设备状态变更通知示例：

```
设备状态变更通知

{
  "resource" : "device.status",
  "event" : "update",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "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" : {
    "last_online_time" : "20151212T121212Z",
    "status" : "ONLINE",
    "status_update_time" : "2015-12-12T12:12:12.815Z"
  }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.2 设备属性上报通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device.property，Event:report）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当设备上报属性数据时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备属性上报通知规则的转发目标决定

请求参数

表 1-297 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device.property。
event	是	String	**参数说明**：订阅的资源事件，取值为report。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DevicePropertyReportNotifyData** object	**参数说明**：推送消息。

表 1-298 DevicePropertyReportNotifyData

参数	是否必选	参数类型	描述
header	是	**NotifyDataHeader** object	**参数说明**：推送消息header。
body	是	**DevicePropertyReport** object	**参数说明**：推送消息body。

表 1-299 NotifyDataHeader

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：应用ID。 最大长度：**256**
device_id	否	String	**参数说明**：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。 最大长度：**256**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**256**
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在注册产品时由物联网平台分配获得。 最大长度：**256**
gateway_id	否	String	**参数说明**：网关ID，用于标识一个网关设备。当设备是直连设备时，gatewayId与设备的deviceId一致。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 最大长度：**256**
tags	否	Array of **TagV5DTO** objects	**参数说明**：要绑定到指定资源的标签列表，标签列表中各项标签键值之间不允许重复，一个资源最多可以绑定10个标签。

表 1-300 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

表 1-301 DevicePropertyReport

参数	是否必选	参数类型	描述
services	是	Array of **DevicePropertyV5** objects	**参数说明**：设备的服务信息列表。

表 1-302 DevicePropertyV5

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID，在设备关联的产品模型中定义。
properties	是	Object	**参数说明**：设备上报的数据。
event_time	是	String	**参数说明**：设备数据上报的时间，格式取决于设备侧上报属性格式，支持的秒级格式：yyyyMMdd'T'HHmmss'Z'，毫秒级格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z'，例如20151212T121212Z或者2020-08-12T12:12:12.333Z。

响应参数

无

请求示例

设备属性上报通知示例：

```
设备属性上报通知

{
  "resource" : "device.property",
  "event" : "report",
```



```
 "event_time" : "20151212T121212Z",
 "event_time_ms" : "2015-12-12T12:12:12.000Z",
 "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
 "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" : {
   "services" : [ {
     "service_id" : "Battery",
     "properties" : {
       "batteryLevel" : 80
     },
     "event_time" : "20151212T121212Z"
   } ]
 }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.3 设备消息状态变更通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device.message.status，Event:update）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当设备消息状态变更时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备消息状态变更通知规则的转发目标决定

请求参数

表 1-303 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device.message.status。
event	是	String	**参数说明**：订阅的资源事件，取值为update。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DeviceMessageStatusUpdateNotifyDataV5** object	**参数说明**：推送消息。

表 1-304 DeviceMessageStatusUpdateNotifyDataV5

参数	是否必选	参数类型	描述
header	是	**NotifyDataHeader** object	**参数说明**：推送消息header。
body	是	**DeviceMessageStatusUpdate** object	**参数说明**：推送消息body。

表 1-305 NotifyDataHeader

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：应用ID。 最大长度：**256**
device_id	否	String	**参数说明**：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。 最大长度：**256**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**256**
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在注册产品时由物联网平台分配获得。 最大长度：**256**
gateway_id	否	String	**参数说明**：网关ID，用于标识一个网关设备。当设备是直连设备时，gatewayId与设备的deviceId一致。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 最大长度：**256**
tags	否	Array of **TagV5DTO** objects	**参数说明**：要绑定到指定资源的标签列表，标签列表中各项标签键值之间不允许重复，一个资源最多可以绑定10个标签。

表 1-306 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

表 1-307 DeviceMessageStatusUpdate

参数	是否必选	参数类型	描述
topic	否	String	**参数说明**：消息上报使用的mqtt topic。
message_id	是	String	**参数说明**：消息的序列号，唯一标识一条消息。
name	否	String	**参数说明**：消息名称。
status	否	String	**参数说明**：设备消息状态，包含：PENDING,DELIVERED,TIMEOUT,FAILED。
error_info	否	**ErrorInfoDTO** object	**参数说明**：消息下发失败信息。
timestamp	否	String	**参数说明**：消息更新时间,格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。

表 1-308 ErrorInfoDTO

参数	是否必选	参数类型	描述
error_code	否	String	**参数说明**：异常信息错误码，包含IOTDA.014016和IOTDA.014112。IOTDA.014016表示设备不在线；IOTDA.014112表示设备没有订阅topic。
error_msg	否	String	**参数说明**：异常信息说明，包含设备不在线和设备没有订阅topic说明。

响应参数

无

请求示例

设备消息状态变更通知示例：

```
设备消息状态变更通知

{
  "resource" : "device.message.status",
  "event" : "update",
  "event_time" : "20151212T121212Z",
```



```
"event_time_ms" : "2015-12-12T12:12:12.000Z",
"request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
"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" : {
  "error_info" : {
    "error_msg" : "Send to device failed, device not subscribe topic.",
    "error_code" : "IOTDA.014112"
  },
  "name" : "name",
  "topic" : "topic",
  "message_id" : "1235",
  "status" : "DELIVERED",
  "timestamp" : "20151212T121212Z"
}
}
```


响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.4 批量任务状态变更通知

功能介绍

应用服务器在调用物联网平台**创建规则触发条件**（Resource:batchtask，Event:update）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当批量任务状态变更时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建批量任务状态变更通知规则的转发目标决定

请求参数

表 1-309 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为batchtask。
event	是	String	**参数说明**：订阅的资源事件，取值为update。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**BatchTaskUpdateNotifyData** object	**参数说明**：设备的自定义字段列表。

表 1-310 BatchTaskUpdateNotifyData

参数	是否必选	参数类型	描述
body	是	**BatchTaskUpdate** object	**参数说明**：推送消息body。

表 1-311 BatchTaskUpdate

参数	是否必选	参数类型	描述
app_id	是	String	**参数说明**：应用ID。

参数	是否必选	参数类型	描述
task_id	是	String	**参数说明**：批量任务ID。调用创建批量任务接口时返回的任务ID。
task_type	是	String	**参数说明**：任务类型。 • firmwareUpgrade：固件升级 • softwareUpgrade：软件升级
status	是	String	**参数说明**：任务状态。 • Waitting 任务正在被等待执行 • Processing 任务正在执行 • Success 任务成功 • PartialSuccess 任务部分成功 • Fail 任务失败 • Stopped 任务被停止
status_desc	是	String	**参数说明**：任务状态描述。

响应参数

无

请求示例

批量任务状态变更通知示例：

批量任务状态变更通知

```
{
  "resource" : "batchtask",
  "event" : "update",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "notify_data" : {
    "body" : {
      "status_desc" : "status_desc",
      "task_id" : "1a7ffc5c-d89c-44dd-8265",
      "task_type" : "softwareUpgrade",
      "app_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
      "status" : "Waitting"
    }
  }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.5 设备消息上报通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device.message，Event:report）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当设备上报消息数据时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备消息上报通知规则的转发目标决定

请求参数

表 1-312 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device.message。
event	是	String	**参数说明**：订阅的资源事件，report。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。

参数	是否必选	参数类型	描述
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DeviceMessageReportNotifyData** object	**参数说明**：设备的自定义字段列表。

表 1-313 DeviceMessageReportNotifyData

参数	是否必选	参数类型	描述
header	是	**NotifyDataHeader** object	**参数说明**：推送消息header。
body	是	**DeviceMessageReport** object	**参数说明**：推送消息body。

表 1-314 NotifyDataHeader

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：应用ID。 最大长度：**256**
device_id	否	String	**参数说明**：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。 最大长度：**256**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**256**
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在注册产品时由物联网平台分配获得。 最大长度：**256**

参数	是否必选	参数类型	描述
gateway_id	否	String	**参数说明**：网关ID，用于标识一个网关设备。当设备是直连设备时，gatewayId与设备的deviceId一致。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 最大长度：**256**
tags	否	Array of **TagV5DTO** objects	**参数说明**：要绑定到指定资源的标签列表，标签列表中各项标签键值之间不允许重复，一个资源最多可以绑定10个标签。

表 1-315 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

表 1-316 DeviceMessageReport

参数	是否必选	参数类型	描述
topic	是	String	**参数说明**：设备上报的mqtt topic。
content	是	Object	**参数说明**：消息内容。
properties	否	**MqttPropertiesDTO** object	**参数说明**：设备携带的属性信息。

表 1-317 MqttPropertiesDTO

参数	是否必选	参数类型	描述
correlation_data	否	String	**参数说明**：MQTT 5.0版本请求和响应模式中的相关数据，可选。设备可以通过该参数配置MQTT协议请求和响应模式中的相关数据。 最大长度：**128**
response_topic	否	String	**参数说明**：MQTT 5.0版本请求和响应模式中的响应主题，可选。设备可以通过该参数配置MQTT协议请求和响应模式中的响应主题。 最大长度：**128**
content_type	否	String	**参数说明**：MQTT 5.0版本有效负载的内容类型，可选。设备可以通过该参数配置MQTT协议有效负载的内容类型。 最大长度：**128**
user_properties	否	Array of **UserPropDTO** objects	**参数说明**：用户自定义属性，可选。设备可以通过该参数配置用户自定义属性。

表 1-318 UserPropDTO

参数	是否必选	参数类型	描述
prop_key	否	String	**参数说明**：用户自定义属性键。 最大长度：**128**
prop_value	否	String	**参数说明**：用户自定义属性值。 最大长度：**128**

响应参数

无

请求示例

设备消息上报通知示例：

```
设备消息上报通知

{
  "resource" : "device.message",
  "event" : "report",
  "event_time" : "20151212T121212Z",
```



```
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "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" : "msg",
    "properties" : {
      "response_topic" : "/device/message/response",
      "content_type" : "text/plain",
      "user_properties" : [ {
        "prop_value" : "propValue1",
        "prop_key" : "propKey1"
      } ],
      "correlation_data" : "messageName"
    }
  }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.6 设备添加通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device，Event:create）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当物联网平台中的设备添加时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备添加通知规则的转发目标决定

请求参数

表 1-319 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device。
event	是	String	**参数说明**：订阅的资源事件，取值为create。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DeviceCreateOrUpdateNotifyData** object	**参数说明**：推送消息。

表 1-320 DeviceCreateOrUpdateNotifyData

参数	是否必选	参数类型	描述
body	是	**QueryDeviceBase** object	**参数说明**：推送消息内容。

表 1-321 QueryDeviceBase

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。 最大长度：**36**

参数	是否必选	参数类型	描述
app_name	否	String	**参数说明**：资源空间名称。
device_id	否	String	**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。 最大长度：**256**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**64**
gateway_id	否	String	**参数说明**：网关ID，用于标识设备所属的父设备，即父设备的设备ID。当设备是直连设备时，gateway_id与设备的device_id一致。当设备是非直连设备时，gateway_id为设备所关联的父设备的device_id。 最大长度：**256**
device_name	否	String	**参数说明**：设备名称。 最大长度：**256**
node_type	否	String	**参数说明**：设备节点类型。 • ENDPOINT：非直连设备。 • GATEWAY：直连设备或网关。 • UNKNOWN：未知。
description	否	String	**参数说明**：设备的描述信息。 最大长度：**2048**
fw_version	否	String	**参数说明**：设备的固件版本。 最大长度：**256**
sw_version	否	String	**参数说明**：设备的软件版本。 最大长度：**256**
device_sdk_version	否	String	**参数说明**：设备的SDK版本。 最大长度：**256**
auth_info	否	**AuthInfo** object	**参数说明**：设备的接入认证信息。
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型。

参数	是否必选	参数类型	描述
product_name	否	String	**参数说明**：设备关联的产品名称。
status	否	String	**参数说明**：设备的状态。 • ONLINE：设备在线。 • OFFLINE：设备离线。 • ABNORMAL：设备异常。 • INACTIVE：设备未激活。 • FREEZED：设备冻结。
create_time	否	String	**参数说明**：在物联网平台注册设备的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
tags	否	Array of **TagV5DTO** objects	**参数说明**：设备的标签列表。
extension_info	否	Object	**参数说明**：设备扩展信息。用户可以自定义任何想要的扩展信息，如果在创建设备时为子设备指定该字段，将会通过MQTT接口“平台通知网关子设备新增“将该信息通知给网关。

表 1-322 AuthInfo

参数	是否必选	参数类型	描述
auth_type	否	String	**参数说明**：鉴权类型。支持密钥认证接入(SECRET)和证书认证接入(CERTIFICATES)两种方式。使用密钥认证接入方式(SECRET)填写secret字段，使用证书认证接入方式(CERTIFICATES)填写fingerprint字段，不填写auth_type默认为密钥认证接入方式(SECRET)
secure_access	否	Boolean	**参数说明**：指设备是否通过安全协议方式接入，默认值为true。 • true：通过安全协议方式接入。 • false：通过非安全协议方式接入。 缺省值：**true**

参数	是否必选	参数类型	描述
timeout	否	Integer	**参数说明**：设备验证码的有效时间，单位：秒，默认值：0。若设备在有效时间内未接入物联网平台并激活，则平台会删除该设备的注册信息。若设置为“0”，则表示设备验证码不会失效（建议填写为“0”）。注意：只有注册设备接口或者修改设备接口修改timeout时返回该参数。 最小值：**0** 最大值：**2147483647** 缺省值：**0**

表 1-323 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

响应参数

无

请求示例

设备添加通知示例：

```
设备添加通知

{
  "resource" : "device",
  "event" : "create",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "notify_data" : {
    "body" : {
      "device_sdk_version" : "C_v0.5.0",
      "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
      "create_time" : "20190303T081011Z",
      "description" : "watermeter device",
      "auth_info" : {
        "auth_type" : "SECRET",
        "secure_access" : true,
        "timeout" : 300
```



```
    },
    "product_name" : "Thermometer",
    "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "sw_version" : "1.1.0",
    "tags" : [ {
      "tag_value" : "testTagValue",
      "tag_key" : "testTagName"
    } ],
    "extension_info" : {
      "aaa" : "xxx",
      "bbb" : 0
    },
    "app_name" : "testAPP01",
    "device_name" : "dianadevice",
    "node_type" : "ENDPOINT",
    "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "fw_version" : "1.1.0",
    "node_id" : "ABC123456789",
    "status" : "INACTIVE"
  }
 }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.7 设备更新通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device，Event:update）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当物联网平台中的设备更新时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备更新通知规则的转发目标决定

请求参数

表 1-324 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device。
event	是	String	**参数说明**：订阅的资源事件，取值为update。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DeviceCreateOrUpdateNotifyData** object	**参数说明**：推送消息。

表 1-325 DeviceCreateOrUpdateNotifyData

参数	是否必选	参数类型	描述
body	是	**QueryDeviceBase** object	**参数说明**：推送消息内容。

表 1-326 QueryDeviceBase

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。 最大长度：**36**

参数	是否必选	参数类型	描述
app_name	否	String	**参数说明**：资源空间名称。
device_id	否	String	**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。 最大长度：**256**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**64**
gateway_id	否	String	**参数说明**：网关ID，用于标识设备所属的父设备，即父设备的设备ID。当设备是直连设备时，gateway_id与设备的device_id一致。当设备是非直连设备时，gateway_id为设备所关联的父设备的device_id。 最大长度：**256**
device_name	否	String	**参数说明**：设备名称。 最大长度：**256**
node_type	否	String	**参数说明**：设备节点类型。 • ENDPOINT：非直连设备。 • GATEWAY：直连设备或网关。 • UNKNOWN：未知。
description	否	String	**参数说明**：设备的描述信息。 最大长度：**2048**
fw_version	否	String	**参数说明**：设备的固件版本。 最大长度：**256**
sw_version	否	String	**参数说明**：设备的软件版本。 最大长度：**256**
device_sdk_version	否	String	**参数说明**：设备的SDK版本。 最大长度：**256**
auth_info	否	**AuthInfo** object	**参数说明**：设备的接入认证信息。
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型。

参数	是否必选	参数类型	描述
product_name	否	String	**参数说明**：设备关联的产品名称。
status	否	String	**参数说明**：设备的状态。 • ONLINE：设备在线。 • OFFLINE：设备离线。 • ABNORMAL：设备异常。 • INACTIVE：设备未激活。 • FREEZED：设备冻结。
create_time	否	String	**参数说明**：在物联网平台注册设备的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
tags	否	Array of **TagV5DTO** objects	**参数说明**：设备的标签列表。
extension_info	否	Object	**参数说明**：设备扩展信息。用户可以自定义任何想要的扩展信息，如果在创建设备时为子设备指定该字段，将会通过MQTT接口“平台通知网关子设备新增“将该信息通知给网关。

表 1-327 AuthInfo

参数	是否必选	参数类型	描述
auth_type	否	String	**参数说明**：鉴权类型。支持密钥认证接入(SECRET)和证书认证接入(CERTIFICATES)两种方式。使用密钥认证接入方式(SECRET)填写secret字段，使用证书认证接入方式(CERTIFICATES)填写fingerprint字段，不填写auth_type默认为密钥认证接入方式(SECRET)
secure_access	否	Boolean	**参数说明**：指设备是否通过安全协议方式接入，默认值为true。 • true：通过安全协议方式接入。 • false：通过非安全协议方式接入。 缺省值：**true**

参数	是否必选	参数类型	描述
timeout	否	Integer	**参数说明**：设备验证码的有效时间，单位：秒，默认值：0。若设备在有效时间内未接入物联网平台并激活，则平台会删除该设备的注册信息。若设置为“0”，则表示设备验证码不会失效（建议填写为“0”）。注意：只有注册设备接口或者修改设备接口修改timeout时返回该参数。 最小值：**0** 最大值：**2147483647** 缺省值：**0**

表 1-328 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

响应参数

无

请求示例

设备更新通知示例：

```
设备更新通知

{
  "resource" : "device",
  "event" : "update",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "notify_data" : {
    "body" : {
      "device_sdk_version" : "C_v0.5.0",
      "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
      "create_time" : "20190303T081011Z",
      "description" : "watermeter device",
      "auth_info" : {
        "auth_type" : "SECRET",
        "secure_access" : true,
        "timeout" : 300
```



```
    },
    "product_name" : "Thermometer",
    "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "sw_version" : "1.1.0",
    "tags" : [ {
      "tag_value" : "testTagValue",
      "tag_key" : "testTagName"
    } ],
    "extension_info" : {
      "aaa" : "xxx",
      "bbb" : 0
    },
    "app_name" : "testAPP01",
    "device_name" : "dianadevice",
    "node_type" : "ENDPOINT",
    "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "fw_version" : "1.1.0",
    "node_id" : "ABC123456789",
    "status" : "INACTIVE"
  }
 }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.8 设备删除通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device，Event:delete）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当物联网平台中的设备删除时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备删除通知规则的转发目标决定

请求参数

表 1-329 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device。
event	是	String	**参数说明**：订阅的资源事件，取值为delete。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DeviceDeleteNotifyData** object	**参数说明**：推送消息。

表 1-330 DeviceDeleteNotifyData

参数	是否必选	参数类型	描述
header	是	**NotifyDataHeader** object	**参数说明**：推送消息header。

表 1-331 NotifyDataHeader

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：应用ID。 最大长度：**256**

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：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。 最大长度：**256**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**256**
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在注册产品时由物联网平台分配获得。 最大长度：**256**
gateway_id	否	String	**参数说明**：网关ID，用于标识一个网关设备。当设备是直连设备时，gatewayId与设备的deviceId一致。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 最大长度：**256**
tags	否	Array of **TagV5DTO** objects	**参数说明**：要绑定到指定资源的标签列表，标签列表中各项标签键值之间不允许重复，一个资源最多可以绑定10个标签。

表 1-332 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

响应参数

无

请求示例

设备删除通知示例：

设备删除通知

```
{
  "resource" : "device",
  "event" : "delete",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "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"
      } ]
    }
  }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.9 产品添加通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:product，Event:create）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当物联网平台中的产品添加时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建产品添加通知规则的转发目标决定

请求参数

表 1-333 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为product。
event	是	String	**参数说明**：订阅的资源事件，取值为create。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**ProductUpdateNotifyData** object	**参数说明**：推送消息。

表 1-334 ProductUpdateNotifyData

参数	是否必选	参数类型	描述
body	是	**Product** object	**参数说明**：推送消息内容。

表 1-335 Product

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。
app_name	否	String	**参数说明**：资源空间名称。

参数	是否必选	参数类型	描述
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。
name	否	String	**参数说明**：产品名称。
device_type	否	String	**参数说明**：设备类型。
protocol_type	否	String	**参数说明**：设备使用的协议类型。取值范围：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF。
data_format	否	String	**参数说明**：设备上报数据的格式，取值范围：json，binary。
manufacturer_name	否	String	**参数说明**：厂商名称。
industry	否	String	**参数说明**：设备所属行业。
description	否	String	**参数说明**：产品的描述信息。
service_capabilities	否	Array of **ServiceCapability** objects	**参数说明**：设备的服务能力列表。
create_time	否	String	**参数说明**：在物联网平台创建产品的时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-336 ServiceCapability

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID。
service_type	是	String	**参数说明**：设备的服务类型。
properties	否	Array of **ServiceProperty** objects	**参数说明**：设备服务支持的属性列表。
commands	否	Array of **ServiceCommand** objects	**参数说明**：设备服务支持的命令列表。
events	否	Array of **ServiceEvent** objects	**参数说明**：设备服务支持的事件列表。

参数	是否必选	参数类型	描述
description	否	String	**参数说明**：设备服务的描述信息。
option	否	String	**参数说明**：指定设备服务是否必选。Master（主服务），Mandatory（必选服务），Optional（可选服务），目前本字段为非功能性字段，仅起到标识作用。默认为Optional（可选服务）。 缺省值：**Optional**

表 1-337 ServiceProperty

参数	是否必选	参数类型	描述
property_name	是	String	**参数说明**：设备属性名称。
required	否	Boolean	**参数说明**：设备属性是否必选。默认为false。 缺省值：**false**
data_type	是	String	**参数说明**：设备属性的数据类型。取值范围：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
enum_list	否	Array of strings	**参数说明**：设备属性的枚举值列表。
min	否	String	**参数说明**：设备属性的最小值。 最小长度：**1** 最大长度：**16**
max	否	String	**参数说明**：设备属性的最大值。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：设备属性的最大长度。
step	否	Double	**参数说明**：设备属性的步长。
unit	否	String	**参数说明**：设备属性的单位。 最大长度：**16**

参数	是否必选	参数类型	描述
method	是	String	**参数说明**：设备属性的访问模式。取值范围：RWE，RW，RE，WE，E，W，R。 • R：属性值可读 • W：属性值可写 • E：属性值可订阅，即属性值变化时上报事件。
description	否	String	**参数说明**：设备属性的描述。
default_value	否	Object	**参数说明**：设备属性的默认值。如果设置了默认值，使用该产品创建设备时，会将该属性的默认值写入到该设备的设备影子预期数据中，待设备上线时将该属性默认值下发给设备。

表 1-338 ServiceCommand

参数	是否必选	参数类型	描述
command_name	是	String	**参数说明**：设备命令名称。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令的参数列表。
responses	否	Array of **ServiceCommandResponse** objects	**参数说明**：设备命令的响应列表。

表 1-339 ServiceCommandResponse

参数	是否必选	参数类型	描述
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令响应的参数列表。
response_name	是	String	**参数说明**：设备命令响应名称。

表 1-340 ServiceEvent

参数	是否必选	参数类型	描述
event_type	是	String	**参数说明**：设备事件类型。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备事件的参数列表。

表 1-341 ServiceCommandPara

参数	是否必选	参数类型	描述
para_name	是	String	**参数说明**：参数的名称。
required	否	Boolean	**参数说明**：参数是否必选。默认为false。 缺省值：**false**
data_type	是	String	**参数说明**：参数的数据类型。取值范围：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
enum_list	否	Array of strings	**参数说明**：参数的枚举值列表。
min	否	String	**参数说明**：参数的最小值。 最小长度：**1** 最大长度：**16**
max	否	String	**参数说明**：参数的最大值。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：参数的最大长度。
step	否	Double	**参数说明**：参数的步长。
unit	否	String	**参数说明**：参数的单位。 最大长度：**16**
description	否	String	**参数说明**：参数的描述。

响应参数

无

请求示例

产品添加通知示例：

产品添加通知


```
{
  "resource" : "product",
  "event" : "create",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "notify_data" : {
    "body" : {
      "app_name" : "testAPP01",
      "protocol_type" : "CoAP",
      "data_format" : "binary",
      "service_capabilities" : [ {
        "service_type" : "temperature",
        "service_id" : "temperature",
        "description" : "temperature",
        "properties" : [ {
          "unit" : "centigrade",
          "min" : "1",
          "method" : "R",
          "max" : "100",
          "data_type" : "decimal",
          "description" : "force",
          "step" : 0.1,
          "default_value" : {
            "color" : "red",
            "size" : 1
          },
          "enum_list" : [ "string" ],
          "required" : true,
          "property_name" : "temperature",
          "max_length" : 100
        } ],
        "commands" : [ {
          "command_name" : "reboot",
          "responses" : [ {
            "response_name" : "ACK",
            "paras" : [ {
              "unit" : "km/h",
              "min" : "1",
              "max" : "100",
              "para_name" : "force",
              "data_type" : "string",
              "description" : "force",
              "step" : 0.1,
              "enum_list" : [ "string" ],
              "required" : false,
              "max_length" : 100
            } ]
          } ],
          "paras" : [ {
            "unit" : "km/h",
            "min" : "1",
            "max" : "100",
            "para_name" : "force",
            "data_type" : "string",
            "description" : "force",
            "step" : 0.1,
            "enum_list" : [ "string" ],
            "required" : false,
            "max_length" : 100
          } ]
        } ],
        "events" : [ {
```

```
        "event_type" : "reboot",
        "paras" : [ {
          "unit" : "km/h",
          "min" : "1",
          "max" : "100",
          "para_name" : "force",
          "data_type" : "string",
          "description" : "force",
          "step" : 0.1,
          "enum_list" : [ "string" ],
          "required" : false,
          "max_length" : 100
        } ]
      } ],
      "option" : "Mandatory"
    } ],
    "create_time" : "20190303T081011Z",
    "product_id" : "5ba24f5ebbe8f56f5a14f605",
    "name" : "Thermometer",
    "description" : "this is a thermometer produced by Huawei",
    "device_type" : "Thermometer",
    "industry" : "smartCity",
    "manufacturer_name" : "ABC",
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
  }
 }
}
```


响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.10 产品更新通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:product，Event:update）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当物联网平台中的产品更新时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建产品更新通知规则的转发目标决定

请求参数

表 1-342 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为product。
event	是	String	**参数说明**：订阅的资源事件，取值为update。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**ProductUpdateNotifyData** object	**参数说明**：推送消息。

表 1-343 ProductUpdateNotifyData

参数	是否必选	参数类型	描述
body	是	**Product** object	**参数说明**：推送消息内容。

表 1-344 Product

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。
app_name	否	String	**参数说明**：资源空间名称。

参数	是否必选	参数类型	描述
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。
name	否	String	**参数说明**：产品名称。
device_type	否	String	**参数说明**：设备类型。
protocol_type	否	String	**参数说明**：设备使用的协议类型。取值范围：MQTT，CoAP，HTTP，HTTPS，Modbus，ONVIF。
data_format	否	String	**参数说明**：设备上报数据的格式，取值范围：json，binary。
manufacturer_name	否	String	**参数说明**：厂商名称。
industry	否	String	**参数说明**：设备所属行业。
description	否	String	**参数说明**：产品的描述信息。
service_capabilities	否	Array of **ServiceCapability** objects	**参数说明**：设备的服务能力列表。
create_time	否	String	**参数说明**：在物联网平台创建产品的时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-345 ServiceCapability

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID。
service_type	是	String	**参数说明**：设备的服务类型。
properties	否	Array of **ServiceProperty** objects	**参数说明**：设备服务支持的属性列表。
commands	否	Array of **ServiceCommand** objects	**参数说明**：设备服务支持的命令列表。
events	否	Array of **ServiceEvent** objects	**参数说明**：设备服务支持的事件列表。

参数	是否必选	参数类型	描述
description	否	String	**参数说明**：设备服务的描述信息。
option	否	String	**参数说明**：指定设备服务是否必选。Master（主服务），Mandatory（必选服务），Optional（可选服务），目前本字段为非功能性字段，仅起到标识作用。默认为Optional（可选服务）。 缺省值：**Optional**

表 1-346 ServiceProperty

参数	是否必选	参数类型	描述
property_name	是	String	**参数说明**：设备属性名称。
required	否	Boolean	**参数说明**：设备属性是否必选。默认为false。 缺省值：**false**
data_type	是	String	**参数说明**：设备属性的数据类型。取值范围：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
enum_list	否	Array of strings	**参数说明**：设备属性的枚举值列表。
min	否	String	**参数说明**：设备属性的最小值。 最小长度：**1** 最大长度：**16**
max	否	String	**参数说明**：设备属性的最大值。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：设备属性的最大长度。
step	否	Double	**参数说明**：设备属性的步长。
unit	否	String	**参数说明**：设备属性的单位。 最大长度：**16**

参数	是否必选	参数类型	描述
method	是	String	**参数说明**：设备属性的访问模式。取值范围：RWE，RW，RE，WE，E，W，R。 • R：属性值可读 • W：属性值可写 • E：属性值可订阅，即属性值变化时上报事件。
description	否	String	**参数说明**：设备属性的描述。
default_value	否	Object	**参数说明**：设备属性的默认值。如果设置了默认值，使用该产品创建设备时，会将该属性的默认值写入到该设备的设备影子预期数据中，待设备上线时将该属性默认值下发给设备。

表 1-347 ServiceCommand

参数	是否必选	参数类型	描述
command_name	是	String	**参数说明**：设备命令名称。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令的参数列表。
responses	否	Array of **ServiceCommandResponse** objects	**参数说明**：设备命令的响应列表。

表 1-348 ServiceCommandResponse

参数	是否必选	参数类型	描述
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备命令响应的参数列表。
response_name	是	String	**参数说明**：设备命令响应名称。

表 1-349 ServiceEvent

参数	是否必选	参数类型	描述
event_type	是	String	**参数说明**：设备事件类型。
paras	否	Array of **ServiceCommandPara** objects	**参数说明**：设备事件的参数列表。

表 1-350 ServiceCommandPara

参数	是否必选	参数类型	描述
para_name	是	String	**参数说明**：参数的名称。
required	否	Boolean	**参数说明**：参数是否必选。默认为false。 缺省值：**false**
data_type	是	String	**参数说明**：参数的数据类型。取值范围：int，long，decimal，string，DateTime，jsonObject，enum，boolean，string list。
enum_list	否	Array of strings	**参数说明**：参数的枚举值列表。
min	否	String	**参数说明**：参数的最小值。 最小长度：**1** 最大长度：**16**
max	否	String	**参数说明**：参数的最大值。 最小长度：**1** 最大长度：**16**
max_length	否	Integer	**参数说明**：参数的最大长度。
step	否	Double	**参数说明**：参数的步长。
unit	否	String	**参数说明**：参数的单位。 最大长度：**16**
description	否	String	**参数说明**：参数的描述。

响应参数

无

请求示例

产品更新通知示例：

```
产品更新通知

{
  "resource" : "product",
  "event" : "update",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "notify_data" : {
    "body" : {
      "app_name" : "testAPP01",
      "protocol_type" : "CoAP",
      "data_format" : "binary",
      "service_capabilities" : [ {
        "service_type" : "temperature",
        "service_id" : "temperature",
        "description" : "temperature",
        "properties" : [ {
          "unit" : "centigrade",
          "min" : "1",
          "method" : "R",
          "max" : "100",
          "data_type" : "decimal",
          "description" : "force",
          "step" : 0.1,
          "default_value" : {
            "color" : "red",
            "size" : 1
          },
          "enum_list" : [ "string" ],
          "required" : true,
          "property_name" : "temperature",
          "max_length" : 100
        } ],
        "commands" : [ {
          "command_name" : "reboot",
          "responses" : [ {
            "response_name" : "ACK",
            "paras" : [ {
              "unit" : "km/h",
              "min" : "1",
              "max" : "100",
              "para_name" : "force",
              "data_type" : "string",
              "description" : "force",
              "step" : 0.1,
              "enum_list" : [ "string" ],
              "required" : false,
              "max_length" : 100
            } ]
          } ],
          "paras" : [ {
            "unit" : "km/h",
            "min" : "1",
            "max" : "100",
            "para_name" : "force",
            "data_type" : "string",
            "description" : "force",
            "step" : 0.1,
            "enum_list" : [ "string" ],
            "required" : false,
            "max_length" : 100
          } ]
        } ],
        "events" : [ {
```



```
        "event_type" : "reboot",
        "paras" : [ {
          "unit" : "km/h",
          "min" : "1",
          "max" : "100",
          "para_name" : "force",
          "data_type" : "string",
          "description" : "force",
          "step" : 0.1,
          "enum_list" : [ "string" ],
          "required" : false,
          "max_length" : 100
        } ]
      } ],
      "option" : "Mandatory"
    } ],
    "create_time" : "20190303T081011Z",
    "product_id" : "5ba24f5ebbe8f56f5a14f605",
    "name" : "Thermometer",
    "description" : "this is a thermometer produced by Huawei",
    "device_type" : "Thermometer",
    "industry" : "smartCity",
    "manufacturer_name" : "ABC",
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
  }
 }
}
```


响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.11 产品删除通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:product，Event:delete）、**创建规则动作**、**修改规则触发条件**接口配置并激活规则后，当物联网平台中的产品删除时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建产品删除通知规则的转发目标决定

请求参数

表 1-351 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为product。
event	是	String	**参数说明**：订阅的资源事件，取值为delete。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串，如：2015-12-12T12:12:12.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**ProductDeleteNotifyData** object	**参数说明**：推送消息。

表 1-352 ProductDeleteNotifyData

参数	是否必选	参数类型	描述
body	是	**DeletedProduct** object	**参数说明**：推送消息内容。

表 1-353 DeletedProduct

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。

参数	是否必选	参数类型	描述
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在物联网平台创建产品后由平台分配获得。
name	否	String	**参数说明**：产品名称。

响应参数

无

请求示例

产品删除通知示例：

```
产品删除通知

{
  "resource" : "product",
  "event" : "delete",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "notify_data" : {
    "body" : {
      "product_id" : "5ba24f5ebbe8f56f5a14f605",
      "name" : "Thermometer",
      "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
    }
  }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.9.12 设备异步命令状态变更通知

功能介绍

应用服务器在调用物联网平台的**创建规则触发条件**（Resource:device.command.status，Event:update）、**创建规则动作**、**修改规则触发**

条件接口配置并激活规则后，当命令状态变更时，物联网平台会向应用服务器推送通知消息。

URI

POST /由应用服务器创建设备异步命令状态变更通知规则的转发目标决定

请求参数

表 1-354 请求 Body 参数

参数	是否必选	参数类型	描述
resource	是	String	**参数说明**：订阅的资源名称，取值为device.command.status。
event	是	String	**参数说明**：订阅的资源事件，取值为update。
event_time	是	String	**参数说明**：资源事件生成时间，格式：yyyyMMdd'T'HHmmss'Z' UTC字符串，如：20151212T121212Z。若需要显示本地时区，您需要自己进行时间转换。
event_time_ms	否	String	**参数说明**：资源事件生成时间，格式：yyyy-MM-dd'T'HH:mm:ss.SSS'Z' UTC字符串。如：2019-03-03T08:10:11.000Z。若需要显示本地时区，您需要自己进行时间转换。
request_id	否	String	**参数说明**：消息ID，由设备侧指定或平台生成，用于跟踪业务流程。
notify_data	是	**DeviceCommandStatusUpdateNotifyDataV5** object	**参数说明**：推送消息。

表 1-355 DeviceCommandStatusUpdateNotifyDataV5

参数	是否必选	参数类型	描述
header	是	**NotifyDataHeader** object	**参数说明**：推送消息header。

参数	是否必选	参数类型	描述
body	是	**DeviceCommandStatusUpdate** object	**参数说明**：推送消息body。

表 1-356 NotifyDataHeader

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：应用ID。 最大长度：**256**
device_id	否	String	**参数说明**：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。 最大长度：**256**
node_id	否	String	**参数说明**：设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。 最大长度：**256**
product_id	否	String	**参数说明**：产品ID，用于唯一标识一个产品，在注册产品时由物联网平台分配获得。 最大长度：**256**
gateway_id	否	String	**参数说明**：网关ID，用于标识一个网关设备。当设备是直连设备时，gatewayId与设备的deviceId一致。当设备是非直连设备时，gatewayId为设备所关联的直连设备（即网关）的deviceId。 最大长度：**256**
tags	否	Array of **TagV5DTO** objects	**参数说明**：要绑定到指定资源的标签列表，标签列表中各项标签键值之间不允许重复，一个资源最多可以绑定10个标签。

表 1-357 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。
tag_value	否	String	**参数说明**：标签值。

表 1-358 DeviceCommandStatusUpdate

参数	是否必选	参数类型	描述
command_id	是	String	**参数说明**：命令id，唯一标识一条命令。
created_time	否	String	**参数说明**：命令的创建时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
sent_time	否	String	**参数说明**：物联网平台发送命令的时间，如果命令是立即下发，则该时间与命令创建时间一致，如果是缓存命令， 则是命令实际下发的时间。"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
delivered_time	否	String	**参数说明**：物联网平台将命令送达到设备的时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。
response_time	否	String	**参数说明**：设备响应命令的时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。

参数	是否必选	参数类型	描述
status	否	String	**参数说明**：下发命令的状态。 ● PENDING表示未下发,在物联网平台缓存着 ● EXPIRED表示命令已经过期，即缓存的时间超过设定的expireTime ● SENT表示命令正在下发 ● DELIVERED表示命令已送达设备 ● SUCCESSFUL表示命令已经成功执行 ● FAILED表示命令执行失败 ● TIMEOUT表示命令下发之后，没有收到设备确认或者响应结果而超时。
result	否	Object	**参数说明**：设备命令执行的详细结果，由设备返回，Json格式。

响应参数

无

请求示例

设备异步命令状态变更通知：

设备异步命令状态变更通知

```
{
  "resource" : "device.command.status",
  "event" : "update",
  "event_time" : "20151212T121212Z",
  "event_time_ms" : "2015-12-12T12:12:12.000Z",
  "request_id" : "3fe58d5e-8697-4849-a165-7db128f7e776",
  "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" : {
      "result" : {
        "key" : "value"
      },
      "created_time" : "20151212T121212Z",
      "sent_time" : "20151212T121212Z",
      "command_id" : "id",
```



```
    "delivered_time" : "20151212T131212Z",
    "response_time" : "20151212T131212Z",
    "status" : "SUCCESSFUL"
   }
  }
}
```

响应示例

无

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.10 设备联动规则

批量任务为应用服务器提供批量处理功能，对接入物联网平台的设备进行批量操作。

1.4.10.1 创建规则

功能介绍

应用服务器可调用此接口在物联网平台创建一条规则。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/rules

表 1-359 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID**。

请求参数

表 1-360 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-361 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	String	**参数说明**：规则名称。 最小长度：**1** 最大长度：**128**
description	否	String	**参数说明**：规则的描述信息。 最大长度：**256**
condition_group	是	**ConditionGroup** object	**参数说明**：规则的条件组，包含简单规则和复杂规则集合。
actions	是	Array of **RuleAction** objects	**参数说明**：规则的动作列表，单个规则最多支持设置10个动作。
rule_type	是	String	**参数说明**：规则的类型。 **取值范围**： • DEVICE_LINKAGE：云端联动规则。 • DEVICE_SIDE：端侧规则。

参数	是否必选	参数类型	描述
status	否	String	**参数说明**：规则的状态，默认值：active。 **取值范围**： • active：激活。 • inactive：未激活。
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的规则归属到哪个资源空间下，否则创建的规则将会归属到**默认资源空间**下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
device_side	否	**DeviceSide** object	**参数说明**：端侧执行下发设备信息，当规则类型为DEVICE_SIDE时，该参数必填。

表 1-362 ConditionGroup

参数	是否必选	参数类型	描述
conditions	否	Array of **RuleCondition** objects	**参数说明**：规则的条件列表，单个规则最多支持设置10个条件。
logic	否	String	**参数说明**：规则条件列表中多个条件之间的逻辑关系，默认值：and。 **取值范围**： • and：逻辑且。 • or：逻辑或。
time_range	否	**TimeRange** object	**参数说明**：规则条件触发的有效时间段。

表 1-363 RuleCondition

参数	是否必选	参数类型	描述
type	是	String	**参数说明**：规则条件的类型。 **取值范围**： • DEVICE_DATA：设备属性数据类型条件。 • SIMPLE_TIMER：简单定时类型条件。 • DAILY_TIMER：每日定时类型条件。 • DEVICE_LINKAGE_STATUS：设备状态类型条件。
device_property_condition	否	**DeviceDataCondition** object	**参数说明**：条件中设备数据类型的信息，当type为DEVICE_DATA时，为必选参数。
simple_timer_condition	否	**SimpleTimerType** object	**参数说明**：条件中简单定时类型的信息，当type为SIMPLE_TIMER时，为必选参数。
daily_timer_condition	否	**DailyTimerType** object	**参数说明**：条件中每日定时类型的信息，当type为DAILY_TIMER时，为必选参数。
device_linkage_status_condition	否	**DeviceLinkageStatusCondition** object	**参数说明**：条件中设备状态类型的信息，当规则类型为DEVICE_LINKAGE且type为DEVICE_LINKAGE_STATUS时，为必选参数。

表 1-364 DeviceDataCondition

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：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。存在该参数时设备属性触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。

参数	是否必选	参数类型	描述
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备属性触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
filters	否	Array of **PropertyFilter** objects	数据过滤条件。

表 1-365 PropertyFilter

参数	是否必选	参数类型	描述
path	是	String	**参数说明**：设备属性的路径信息，格式：service_id/DataProperty，例如门磁状态为“DoorWindow/status”。 最大长度：**128**
operator	是	String	**参数说明**：数据比较的操作符。 **取值范围**：当前支持的操作符有：>，<，>=，<=，=，in:表示在指定值中匹配和between:表示数值区间。
value	否	String	**参数说明**：数据比较表达式的右值。与数据比较操作符between联用时，右值表示最小值和最大值，用逗号隔开，如“20,30”表示大于等于20小于30。 最大长度：**64**
in_values	否	Array of strings	**参数说明**：当operator为in时该字段必填，使用该字段传递比较表达式右值，上限为20个。 最大长度：**128**
strategy	否	**Strategy** object	**参数说明**：规则条件的处理策略，用于确定规则是否判断上次数据是否满足条件。当rule_type为DEVICE_LINKAGE时，该参数值不能为空。端侧执行不支持该字段。

表 1-366 Strategy

参数	是否必选	参数类型	描述
trigger	否	String	**参数说明**：规则条件触发的判断策略，默认为pulse。 **取值范围**： • pulse：设备上报的数据满足条件则触发，不判断上一次上报的数据。 • reverse：设备上一次上报的数据不满足条件，本次上报的数据满足条件则触发。
event_valid_time	否	Integer	**参数说明**：设备数据的有效时间，单位为秒，设备数据的产生时间以上报数据中的eventTime为基准。 最小值：**-1**

表 1-367 SimpleTimerType

参数	是否必选	参数类型	描述
start_time	是	String	**参数说明**：规则触发的开始时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。 最大长度：**128**
repeat_interval	是	Integer	**参数说明**：规则触发的重复时间间隔，单位为秒。 最小值：**1** 最大值：**31536000**
repeat_count	是	Integer	**参数说明**：规则触发的重复次数。 最小值：**1** 最大值：**9999**

表 1-368 DailyTimerType

参数	是否必选	参数类型	描述
time	是	String	**参数说明**：规则触发的时间，格式：HH:MM。 最大长度：**128**

参数	是否必选	参数类型	描述
days_of_week	否	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。 最大长度：**128**

表 1-369 DeviceLinkageStatusCondition

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：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。存在该参数时设备状态触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备状态触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
status_list	否	Array of strings	**参数说明**：状态列表，设备状态条件携带该参数。 **取值范围**： • ONLINE：设备上线 • OFFLINE：设备下线 最大长度：**128**
duration	否	Integer	**持续时长**：设备状态持续时长，取值范围: 0-60(分钟)。 最小值：**0** 最大值：**60** 缺省值：**0**

表 1-370 TimeRange

参数	是否必选	参数类型	描述
start_time	是	String	**参数说明**：规则条件触发的开始时间，格式：HH:mm。
end_time	是	String	**参数说明**：规则条件触发的结束时间，格式：HH:mm。若结束时间与开始时间一致，则时间为全天。
days_of_week	否	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。星期列表中的日期为开始时间的日期。

表 1-371 RuleAction

参数	是否必选	参数类型	描述
type	是	String	**参数说明**：规则动作的类型，端侧执行只支持下发设备命令消息类型。 **取值范围**： • DEVICE_CMD：下发设备命令消息类型。 • SMN_FORWARDING：发送SMN消息类型。 • DEVICE_ALARM：上报设备告警消息类型。当选择该类型时，condition中必须有DEVICE_DATA条件类型。该类型动作只能唯一。
device_command	否	**ActionDeviceCommand** object	下发设备命令消息内容。当type为DEVICE_CMD时，必填。
smn_forwarding	否	**ActionSmnForwarding** object	发送给SMN消息结构。当规则类型为DEVICE_LINKAGE且type为SMN_FORWARDING时，必填。
device_alarm	否	**ActionDeviceAlarm** object	上报设备告警消息内容。当规则类型为DEVICE_LINKAGE且type为DEVICE_ALARM时，必填。

表 1-372 ActionDeviceCommand

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：下发命令的设备ID。 • 当创建设备数据规则时，若device_id为空，则命令下发给触发条件的设备。 • 当创建定时规则时，不允许为空。 **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。
cmd	是	**CMD** object	**参数说明**：下发的命令信息。

表 1-373 CMD

参数	是否必选	参数类型	描述
command_name	是	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。 最大长度：**128**

参数	是否必选	参数类型	描述
command_body	是	Object	**参数说明**：设备命令参数，Json格式。 使用LWM2M协议设备命令示例：{"value":"1"}，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。 使用MQTT协议设备命令示例：{"header": {"mode": "ACK","from": "/users/testUser","method": "SET_TEMPERATURE_READ_PERIOD","to":"/devices/{device_id}/services/{service_id}"},"body": {"value" : "1"}}。 ● mode：必选，设备收到命令后是否需要回复确认消息，默认为ACK模式。ACK表示需要回复确认消息，NOACK表示不需要回复确认消息，其它值无效。 ● from：可选，命令发送方的地址。App发起请求时格式为/users/{userId}，应用服务器发起请求时格式为/{serviceName}，物联网平台发起请求时格式为/cloud/{serviceName}。 ● to：可选，命令接收方的地址，格式为/devices/{device_id}/services/{service_id}。 ● method：可选，产品模型中定义的命令名称。 ● body：可选，命令的消息体，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。具体格式需要应用和设备约定。
service_id	是	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。 最大长度：**64**

参数	是否必选	参数类型	描述
buffer_timeout	否	Integer	**参数说明**：设备命令的缓存时间，单位为秒，表示物联网平台在把命令下发给设备前缓存命令的有效时间，超过这个时间后命令将不再下发，默认值为172800s（48小时）。 如果buffer_timeout设置为0，则无论物联网平台上设置的命令下发模式是什么，该命令都会立即下发给设备。 最小值：**0** 最大值：**31536000** 缺省值：**172800**
response_timeout	否	Integer	**参数说明**：命令响应的有效时间，单位为秒，表示设备接收到命令后，在response_timeout时间内响应有效，超过这个时间未收到命令的响应，则认为命令响应超时，默认值为1800s。 最小值：**1** 最大值：**31536000** 缺省值：**1800**
mode	否	String	**参数说明**：设备命令的下发模式，仅当buffer_timeout的值大于0时有效。 ● ACTIVE：主动模式，物联网平台主动将命令下发给设备。 ● PASSIVE：被动模式，物联网平台创建设备命令后，会直接缓存命令。等到设备再次上线或者上报上一条命令的执行结果后才下发命令。

表 1-374 ActionSmnForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：SMN服务对应的region区域。 最大长度：**256**
project_id	是	String	**参数说明**：SMN服务对应的projectId信息。

参数	是否必选	参数类型	描述
theme_name	是	String	**参数说明**：SMN服务对应的主题名称。 最大长度：**256**
topic_urn	是	String	**参数说明**：SMN服务对应的topic的主题URN。 最大长度：**256**
message_content	否	String	**参数说明**：短信或邮件的内容。 最大长度：**1024**
message_template_name	否	String	**参数说明**：SMN服务对应的模板名称。
message_title	是	String	**参数说明**：短信或邮件的主题。最大长度支持UTF-8编码后的521个字节。

表 1-375 ActionDeviceAlarm

参数	是否必选	参数类型	描述
name	是	String	**参数说明**：告警名称。 最大长度：**256**
alarm_status	是	String	**参数说明**：告警状态。 **取值范围**： ● fault：上报告警。 ● recovery：恢复告警。
severity	是	String	**参数说明**：告警级别。 **取值范围**：warning（警告）、minor（一般）、major（严重）和critical（致命）。
dimension	否	String	**参数说明**：告警维度，与告警名称和告警级别组合起来共同标识一条告警，默认不携带该字段为用户维度告警，支持设备维度和资源空间维度告警。 **取值范围**： ● device：设备维度。 ● app：资源空间维度。
description	否	String	**参数说明**：告警的描述信息。 最大长度：**256**

表 1-376 DeviceSide

参数	是否必选	参数类型	描述
device_ids	否	Array of strings	**参数说明**：端侧执行下发的目标设备ID列表。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。 最大长度：**128**

响应参数

状态码： 201

表 1-377 响应 Body 参数

参数	参数类型	描述
rule_id	String	规则id。 最大长度：**128**
name	String	规则名称。 最小长度：**1** 最大长度：**128**
description	String	规则的描述信息。 最大长度：**256**
condition_group	**ConditionGroup** object	规则的条件组，包含简单规则和复杂规则集合。
actions	Array of **RuleAction** objects	规则的动作列表，单个规则最多支持设置10个动作。
rule_type	String	规则的类型 • DEVICE_LINKAGE：云端联动规则。 • DEVICE_SIDE：端侧规则。
status	String	规则的状态，默认值：active。 • active：激活。 • inactive：未激活。
app_id	String	资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的规则归属到哪个资源空间下，否则创建的规则将会归属到**默认资源空间**下。
edge_node_ids	Array of strings	归属边缘侧节点设备ID列表。

参数	参数类型	描述
last_update_time	String	规则最后更新时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。
device_side	**DeviceSide** object	**参数说明**：端侧执行下发设备信息，当规则类型为DEVICE_SIDE时，该参数必填。

表 1-378 ConditionGroup

参数	参数类型	描述
conditions	Array of **RuleCondition** objects	**参数说明**：规则的条件列表，单个规则最多支持设置10个条件。
logic	String	**参数说明**：规则条件列表中多个条件之间的逻辑关系，默认值：and。 **取值范围**： • and：逻辑且。 • or：逻辑或。
time_range	**TimeRange** object	**参数说明**：规则条件触发的有效时间段。

表 1-379 RuleCondition

参数	参数类型	描述
type	String	**参数说明**：规则条件的类型。 **取值范围**： • DEVICE_DATA：设备属性数据类型条件。 • SIMPLE_TIMER：简单定时类型条件。 • DAILY_TIMER：每日定时类型条件。 • DEVICE_LINKAGE_STATUS：设备状态类型条件。
device_property_condition	**DeviceDataCondition** object	**参数说明**：条件中设备数据类型的信息，当type为DEVICE_DATA时，为必选参数。
simple_timer_condition	**SimpleTimerType** object	**参数说明**：条件中简单定时类型的信息，当type为SIMPLE_TIMER时，为必选参数。
daily_timer_condition	**DailyTimerType** object	**参数说明**：条件中每日定时类型的信息，当type为DAILY_TIMER时，为必选参数。
device_linkage_status_condition	**DeviceLinkageStatusCondition** object	**参数说明**：条件中设备状态类型的信息，当规则类型为DEVICE_LINKAGE且type为DEVICE_LINKAGE_STATUS时，为必选参数。

表 1-380 DeviceDataCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备属性触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备属性触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
filters	Array of **PropertyFilter** objects	数据过滤条件。

表 1-381 PropertyFilter

参数	参数类型	描述
path	String	**参数说明**：设备属性的路径信息，格式：service_id/DataProperty，例如门磁状态为“DoorWindow/status”。 最大长度：**128**
operator	String	**参数说明**：数据比较的操作符。 **取值范围**：当前支持的操作符有：>，<，>=，<=，=，in:表示在指定值中匹配和between:表示数值区间。
value	String	**参数说明**：数据比较表达式的右值。与数据比较操作符between联用时，右值表示最小值和最大值，用逗号隔开，如“20,30”表示大于等于20小于30。 最大长度：**64**
in_values	Array of strings	**参数说明**：当operator为in时该字段必填，使用该字段传递比较表达式右值，上限为20个。 最大长度：**128**
strategy	**Strategy** object	**参数说明**：规则条件的处理策略，用于确定规则是否判断上次数据是否满足条件。当rule_type为DEVICE_LINKAGE时，该参数值不能为空。端侧执行不支持该字段。

表 1-382 Strategy

参数	参数类型	描述
trigger	String	**参数说明**：规则条件触发的判断策略，默认为pulse。 **取值范围**： ● pulse：设备上报的数据满足条件则触发，不判断上一次上报的数据。 ● reverse：设备上一次上报的数据不满足条件，本次上报的数据满足条件则触发。
event_valid_time	Integer	**参数说明**：设备数据的有效时间，单位为秒，设备数据的产生时间以上报数据中的eventTime为基准。 最小值：**-1**

表 1-383 SimpleTimerType

参数	参数类型	描述
start_time	String	**参数说明**：规则触发的开始时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。 最大长度：**128**
repeat_interval	Integer	**参数说明**：规则触发的重复时间间隔，单位为秒。 最小值：**1** 最大值：**31536000**
repeat_count	Integer	**参数说明**：规则触发的重复次数。 最小值：**1** 最大值：**9999**

表 1-384 DailyTimerType

参数	参数类型	描述
time	String	**参数说明**：规则触发的时间，格式：HH:MM。 最大长度：**128**
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。 最大长度：**128**

表 1-385 DeviceLinkageStatusCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备状态触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备状态触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
status_list	Array of strings	**参数说明**：状态列表，设备状态条件携带该参数。 **取值范围**： ● ONLINE：设备上线 ● OFFLINE：设备下线 最大长度：**128**
duration	Integer	**持续时长**：设备状态持续时长，取值范围:0-60(分钟)。 最小值：**0** 最大值：**60** 缺省值：**0**

表 1-386 TimeRange

参数	参数类型	描述
start_time	String	**参数说明**：规则条件触发的开始时间，格式：HH:mm。
end_time	String	**参数说明**：规则条件触发的结束时间，格式：HH:mm。若结束时间与开始时间一致，则时间为全天。
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。星期列表中的日期为开始时间的日期。

表 1-387 RuleAction

参数	参数类型	描述
type	String	**参数说明**：规则动作的类型，端侧执行只支持下发设备命令消息类型。 **取值范围**： • DEVICE_CMD：下发设备命令消息类型。 • SMN_FORWARDING：发送SMN消息类型。 • DEVICE_ALARM：上报设备告警消息类型。当选择该类型时，condition中必须有DEVICE_DATA条件类型。该类型动作只能唯一。
device_command	**ActionDeviceCommand** object	下发设备命令消息内容。当type为DEVICE_CMD时，必填。
smn_forwarding	**ActionSmnForwarding** object	发送给SMN消息结构。当规则类型为DEVICE_LINKAGE且type为SMN_FORWARDING时，必填。
device_alarm	**ActionDeviceAlarm** object	上报设备告警消息内容。当规则类型为DEVICE_LINKAGE且type为DEVICE_ALARM时，必填。

表 1-388 ActionDeviceCommand

参数	参数类型	描述
device_id	String	**参数说明**：下发命令的设备ID。 • 当创建设备数据规则时，若device_id为空，则命令下发给触发条件的设备。 • 当创建定时规则时，不允许为空。 **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。
cmd	**CMD** object	**参数说明**：下发的命令信息。

表 1-389 CMD

参数	参数类型	描述
command_name	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。 最大长度：**128**

参数	参数类型	描述
command_body	Object	**参数说明**：设备命令参数，Json格式。 使用LWM2M协议设备命令示例：{"value":"1"}，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。 使用MQTT协议设备命令示例：{"header": {"mode": "ACK","from": "/users/testUser","method": "SET_TEMPERATURE_READ_PERIOD","to":"/devices/{device_id}/services/{service_id}"},"body": {"value" : "1"}}。 ● mode：必选，设备收到命令后是否需要回复确认消息，默认为ACK模式。ACK表示需要回复确认消息，NOACK表示不需要回复确认消息，其它值无效。 ● from：可选，命令发送方的地址。App发起请求时格式为/users/{userId} ，应用服务器发起请求时格式为/{serviceName}，物联网平台发起请求时格式为/cloud/{serviceName}。 ● to：可选，命令接收方的地址，格式为/devices/{device_id}/services/{service_id}。 ● method：可选，产品模型中定义的命令名称。 ● body：可选，命令的消息体，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。具体格式需要应用和设备约定。
service_id	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。 最大长度：**64**
buffer_timeout	Integer	**参数说明**：设备命令的缓存时间，单位为秒，表示物联网平台在把命令下发给设备前缓存命令的有效时间，超过这个时间后命令将不再下发，默认值为172800s（48小时）。 如果buffer_timeout设置为0，则无论物联网平台上设置的命令下发模式是什么，该命令都会立即下发给设备。 最小值：**0** 最大值：**31536000** 缺省值：**172800**

参数	参数类型	描述
response_timeout	Integer	**参数说明**：命令响应的有效时间，单位为秒，表示设备接收到命令后，在response_timeout时间内响应有效，超过这个时间未收到命令的响应，则认为命令响应超时，默认值为1800s。 最小值：**1** 最大值：**31536000** 缺省值：**1800**
mode	String	**参数说明**：设备命令的下发模式，仅当buffer_timeout的值大于0时有效。 ● ACTIVE：主动模式，物联网平台主动将命令下发给设备。 ● PASSIVE：被动模式，物联网平台创建设备命令后，会直接缓存命令。等到设备再次上线或者上报上一条命令的执行结果后才下发命令。

表 1-390 ActionSmnForwarding

参数	参数类型	描述
region_name	String	**参数说明**：SMN服务对应的region区域。 最大长度：**256**
project_id	String	**参数说明**：SMN服务对应的projectId信息。
theme_name	String	**参数说明**：SMN服务对应的主题名称。 最大长度：**256**
topic_urn	String	**参数说明**：SMN服务对应的topic的主题URN。 最大长度：**256**
message_content	String	**参数说明**：短信或邮件的内容。 最大长度：**1024**
message_template_name	String	**参数说明**：SMN服务对应的模板名称。
message_title	String	**参数说明**：短信或邮件的主题。最大长度支持UTF-8编码后的521个字节。

表 1-391 ActionDeviceAlarm

参数	参数类型	描述
name	String	**参数说明**：告警名称。 最大长度：**256**

参数	参数类型	描述
alarm_status	String	**参数说明**：告警状态。 **取值范围**： ● fault：上报告警。 ● recovery：恢复告警。
severity	String	**参数说明**：告警级别。 **取值范围**：warning（警告）、minor（一般）、major（严重）和critical（致命）。
dimension	String	**参数说明**：告警维度，与告警名称和告警级别组合起来共同标识一条告警，默认不携带该字段为用户维度告警，支持设备维度和资源空间维度告警。 **取值范围**： ● device：设备维度。 ● app：资源空间维度。
description	String	**参数说明**：告警的描述信息。 最大长度：**256**

表 1-392 DeviceSide

参数	参数类型	描述
device_ids	Array of strings	**参数说明**：端侧执行下发的目标设备ID列表。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。 最大长度：**128**

请求示例

- 创建云端规则，设备下线触发告警。

```
POST https://{endpoint}/v5/iot/{project_id}/rules

{
  "name" : "openLight",
  "description" : "string",
  "condition_group" : {
    "time_range" : {
      "days_of_week" : "2,3,4,5,6",
      "start_time" : "18:00",
      "end_time" : "23:00"
    },
    "logic" : "or",
    "conditions" : [ {
      "device_linkage_status_condition" : {
        "device_id" : "07b69d78-c716-4be6-9545-869920738397",
        "status_list" : [ "OFFLINE" ],
        "duration" : 0
      },
      "type" : "DEVICE_LINKAGE_STATUS"
    } ]
  },
  "actions" : [ {
```



```
  "device_alarm" : {
    "severity" : "warning",
    "alarm_status" : "fault",
    "name" : "设备下线",
    "description" : "设备下线"
  },
  "type" : "DEVICE_ALARM"
 } ],
 "rule_type" : "DEVICE_LINKAGE",
 "status" : "active",
 "app_id" : "string"
}
```

- 创建云端规则，属性上报触发命令下发。

```
POST https://{endpoint}/v5/iot/{project_id}/rules

{
 "name" : "openLight",
 "description" : "string",
 "condition_group" : {
  "time_range" : {
    "days_of_week" : "2,3,4,5,6",
    "start_time" : "18:00",
    "end_time" : "23:00"
  },
  "logic" : "or",
  "conditions" : [ {
    "device_property_condition" : {
      "device_id" : "07b69d78-c716-4be6-9545-869920738397",
      "filters" : [ {
        "path" : "StreetLight/visibility",
        "strategy" : {
          "trigger" : "reverse",
          "event_valid_time" : 300
        },
        "value" : "30",
        "operator" : "<"
      } ]
    },
    "type" : "DEVICE_DATA"
  } ]
 },
 "actions" : [ {
  "device_command" : {
    "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
    "cmd" : {
      "buffer_timeout" : 0,
      "mode" : "ACTIVE",
      "service_id" : "Switch",
      "command_name" : "SET_LIGHT_SWITCH",
      "command_body" : {
        "value" : 0
      }
    }
  },
  "type" : "DEVICE_CMD"
 } ],
 "rule_type" : "DEVICE_LINKAGE",
 "status" : "active",
 "app_id" : "string"
}
```

- 创建端测规则，属性上报触发命令下发。

```
POST https://{endpoint}/v5/iot/{project_id}/rules

{
 "name" : "openLight",
 "description" : "string",
 "condition_group" : {
  "time_range" : {
```



```
      "days_of_week" : "2,3,4,5,6",
      "start_time" : "18:00",
      "end_time" : "23:00"
    },
    "logic" : "or",
    "conditions" : [ {
      "device_property_condition" : {
        "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
        "filters" : [ {
          "path" : "StreetLight/visibility",
          "strategy" : {
            "trigger" : "reverse",
            "event_valid_time" : 300
          },
          "value" : "30",
          "operator" : "<"
        } ]
      },
      "type" : "DEVICE_DATA"
    } ]
  },
  "actions" : [ {
    "device_command" : {
      "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
      "cmd" : {
        "buffer_timeout" : 0,
        "mode" : "ACTIVE",
        "service_id" : "Switch",
        "command_name" : "SET_LIGHT_SWITCH",
        "command_body" : {
          "value" : 0
        }
      }
    },
    "type" : "DEVICE_CMD"
  } ],
  "rule_type" : "DEVICE_SIDE",
  "device_side" : {
    "device_ids" : [ "3a9e52d9-3ebf-4985-89e9-6d2396748a2f" ]
  },
  "status" : "active",
  "app_id" : "string"
}
```

响应示例

状态码： 201

Created

```
{
  "rule_id" : "5eb3628d017d9105d0cf9aec",
  "rule_type" : "DEVICE_LINKAGE",
  "name" : "openLight",
  "status" : "active",
  "condition_group" : {
    "conditions" : [ {
      "type" : "DEVICE_DATA",
      "device_property_condition" : {
        "device_id" : "07b69d78-c716-4be6-9545-869920738397",
        "filters" : [ {
          "path" : "StreetLight/visibility",
          "operator" : "<",
          "value" : "30",
          "strategy" : {
            "trigger" : "reverse",
            "event_valid_time" : 300
          }
        } ]
```



```
    }
   } ]
 },
 "actions" : [ {
   "type" : "DEVICE_CMD",
   "device_command" : {
     "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
     "cmd" : {
       "command_name" : "SET_LIGHT_SWITCH",
       "command_body" : {
         "value" : 0
       },
       "service_id" : "Switch",
       "buffer_timeout" : 0,
       "response_timeout" : 1800
     }
   }
 } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建云端规则，设备下线触发告警。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
```

```
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRuleRequest request = new CreateRuleRequest();
        Rule body = new Rule();
        ActionDeviceAlarm deviceAlarmActions = new ActionDeviceAlarm();
        deviceAlarmActions.withName("设备下线")
            .withAlarmStatus("fault")
            .withSeverity("warning")
            .withDescription("设备下线");
        List<RuleAction> listbodyActions = new ArrayList<>();
        listbodyActions.add(
            new RuleAction()
                .withType("DEVICE_ALARM")
                .withDeviceAlarm(deviceAlarmActions)
        );
        TimeRange timeRangeConditionGroup = new TimeRange();
        timeRangeConditionGroup.withStartTime("18:00")
            .withEndTime("23:00")
            .withDaysOfWeek("2,3,4,5,6");
        List<String> listDeviceLinkageStatusConditionStatusList = new ArrayList<>();
        listDeviceLinkageStatusConditionStatusList.add("OFFLINE");
        DeviceLinkageStatusCondition deviceLinkageStatusConditionConditions = new
DeviceLinkageStatusCondition();
        deviceLinkageStatusConditionConditions.withDeviceId("07b69d78-
c716-4be6-9545-869920738397")
            .withStatusList(listDeviceLinkageStatusConditionStatusList)
            .withDuration(0);
        List<RuleCondition> listConditionGroupConditions = new ArrayList<>();
        listConditionGroupConditions.add(
            new RuleCondition()
                .withType("DEVICE_LINKAGE_STATUS")
                .withDeviceLinkageStatusCondition(deviceLinkageStatusConditionConditions)
        );
        ConditionGroup conditionGroupbody = new ConditionGroup();
        conditionGroupbody.withConditions(listConditionGroupConditions)
            .withLogic("or")
            .withTimeRange(timeRangeConditionGroup);
        body.withAppId("string");
        body.withStatus("active");
        body.withRuleType("DEVICE_LINKAGE");
        body.withActions(listbodyActions);
        body.withConditionGroup(conditionGroupbody);
        body.withDescription("string");
        body.withName("openLight");
        request.withBody(body);
        try {
            CreateRuleResponse response = client.createRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建云端规则，属性上报触发命令下发。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
```

```
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRuleRequest request = new CreateRuleRequest();
        Rule body = new Rule();
        Cmd cmdDeviceCommand = new Cmd();
        cmdDeviceCommand.withCommandName("SET_LIGHT_SWITCH")
            .withCommandBody("{\"value\":0}")
            .withServiceId("Switch")
            .withBufferTimeout(0)
            .withMode("ACTIVE");
        ActionDeviceCommand deviceCommandActions = new ActionDeviceCommand();
        deviceCommandActions.withDeviceId("3a9e52d9-3ebf-4985-89e9-6d2396748a2f")
            .withCmd(cmdDeviceCommand);
        List listbodyActions = new ArrayList<>();
        listbodyActions.add(
            new RuleAction()
                .withType("DEVICE_CMD")
                .withDeviceCommand(deviceCommandActions)
        );
        TimeRange timeRangeConditionGroup = new TimeRange();
        timeRangeConditionGroup.withStartTime("18:00")
            .withEndTime("23:00")
            .withDaysOfWeek("2,3,4,5,6");
        Strategy strategyFilters = new Strategy();
        strategyFilters.withTrigger("reverse")
            .withEventValidTime(300);
        List listDevicePropertyConditionFilters = new ArrayList<>();
        listDevicePropertyConditionFilters.add(
            new PropertyFilter()
                .withPath("StreetLight/visibility")
                .withOperator("<")
                .withValue("30")
                .withStrategy(strategyFilters)
        );
        DeviceDataCondition devicePropertyConditionConditions = new DeviceDataCondition();
        devicePropertyConditionConditions.withDeviceId("07b69d78-c716-4be6-9545-869920738397")
```

```
            .withFilters(listDevicePropertyConditionFilters);
        List listConditionGroupConditions = new ArrayList<>();
        listConditionGroupConditions.add(
            new RuleCondition()
                .withType("DEVICE_DATA")
                .withDevicePropertyCondition(devicePropertyConditionConditions)
        );
        ConditionGroup conditionGroupbody = new ConditionGroup();
        conditionGroupbody.withConditions(listConditionGroupConditions)
            .withLogic("or")
            .withTimeRange(timeRangeConditionGroup);
        body.withAppId("string");
        body.withStatus("active");
        body.withRuleType("DEVICE_LINKAGE");
        body.withActions(listbodyActions);
        body.withConditionGroup(conditionGroupbody);
        body.withDescription("string");
        body.withName("openLight");
        request.withBody(body);
        try {
            CreateRuleResponse response = client.createRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建端测规则，属性上报触发命令下发。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
```

```
            .withAk(ak)
            .withSk(sk);

    IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
    CreateRuleRequest request = new CreateRuleRequest();
    Rule body = new Rule();
    List<String> listDeviceSideDeviceIds = new ArrayList<>();
    listDeviceSideDeviceIds.add("3a9e52d9-3ebf-4985-89e9-6d2396748a2f");
    DeviceSide deviceSidebody = new DeviceSide();
    deviceSidebody.withDeviceIds(listDeviceSideDeviceIds);
    Cmd cmdDeviceCommand = new Cmd();
    cmdDeviceCommand.withCommandName("SET_LIGHT_SWITCH")
        .withCommandBody("{\"value\":0}")
        .withServiceId("Switch")
        .withBufferTimeout(0)
        .withMode("ACTIVE");
    ActionDeviceCommand deviceCommandActions = new ActionDeviceCommand();
    deviceCommandActions.withDeviceId("3a9e52d9-3ebf-4985-89e9-6d2396748a2f")
        .withCmd(cmdDeviceCommand);
    List<RuleAction> listbodyActions = new ArrayList<>();
    listbodyActions.add(
        new RuleAction()
            .withType("DEVICE_CMD")
            .withDeviceCommand(deviceCommandActions)
    );
    TimeRange timeRangeConditionGroup = new TimeRange();
    timeRangeConditionGroup.withStartTime("18:00")
        .withEndTime("23:00")
        .withDaysOfWeek("2,3,4,5,6");
    Strategy strategyFilters = new Strategy();
    strategyFilters.withTrigger("reverse")
        .withEventValidTime(300);
    List<PropertyFilter> listDevicePropertyConditionFilters = new ArrayList<>();
    listDevicePropertyConditionFilters.add(
        new PropertyFilter()
            .withPath("StreetLight/visibility")
            .withOperator("<")
            .withValue("30")
            .withStrategy(strategyFilters)
    );
    DeviceDataCondition devicePropertyConditionConditions = new DeviceDataCondition();
    devicePropertyConditionConditions.withDeviceId("3a9e52d9-3ebf-4985-89e9-6d2396748a2f")
        .withFilters(listDevicePropertyConditionFilters);
    List<RuleCondition> listConditionGroupConditions = new ArrayList<>();
    listConditionGroupConditions.add(
        new RuleCondition()
            .withType("DEVICE_DATA")
            .withDevicePropertyCondition(devicePropertyConditionConditions)
    );
    ConditionGroup conditionGroupbody = new ConditionGroup();
    conditionGroupbody.withConditions(listConditionGroupConditions)
        .withLogic("or")
        .withTimeRange(timeRangeConditionGroup);
    body.withDeviceSide(deviceSidebody);
    body.withAppId("string");
    body.withStatus("active");
    body.withRuleType("DEVICE_SIDE");
    body.withActions(listbodyActions);
    body.withConditionGroup(conditionGroupbody);
    body.withDescription("string");
    body.withName("openLight");
    request.withBody(body);
    try {
        CreateRuleResponse response = client.createRule(request);
```



```
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建云端规则，设备下线触发告警。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRuleRequest()
        deviceAlarmActions = ActionDeviceAlarm(
            name="设备下线",
            alarm_status="fault",
            severity="warning",
            description="设备下线"
        )
        listActionsbody = [
            RuleAction(
                type="DEVICE_ALARM",
                device_alarm=deviceAlarmActions
            )
        ]
        timeRangeConditionGroup = TimeRange(
            start_time="18:00",
            end_time="23:00",
            days_of_week="2,3,4,5,6"
        )
        listStatusListDeviceLinkageStatusCondition = [
```

```
            "OFFLINE"
        ]
        deviceLinkageStatusConditionConditions = DeviceLinkageStatusCondition(
            device_id="07b69d78-c716-4be6-9545-869920738397",
            status_list=listStatusListDeviceLinkageStatusCondition,
            duration=0
        )
        listConditionsConditionGroup = [
            RuleCondition(
                type="DEVICE_LINKAGE_STATUS",
                device_linkage_status_condition=deviceLinkageStatusConditionConditions
            )
        ]
        conditionGroupbody = ConditionGroup(
            conditions=listConditionsConditionGroup,
            logic="or",
            time_range=timeRangeConditionGroup
        )
        request.body = Rule(
            app_id="string",
            status="active",
            rule_type="DEVICE_LINKAGE",
            actions=listActionsbody,
            condition_group=conditionGroupbody,
            description="string",
            name="openLight"
        )
        response = client.create_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建云端规则，属性上报触发命令下发。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRuleRequest()
        cmdDeviceCommand = Cmd(
```

```
        command_name="SET_LIGHT_SWITCH",
        command_body="{\"value\":0}",
        service_id="Switch",
        buffer_timeout=0,
        mode="ACTIVE"
    )
    deviceCommandActions = ActionDeviceCommand(
        device_id="3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
        cmd=cmdDeviceCommand
    )
    listActionsbody = [
        RuleAction(
            type="DEVICE_CMD",
            device_command=deviceCommandActions
        )
    ]
    timeRangeConditionGroup = TimeRange(
        start_time="18:00",
        end_time="23:00",
        days_of_week="2,3,4,5,6"
    )
    strategyFilters = Strategy(
        trigger="reverse",
        event_valid_time=300
    )
    listFiltersDevicePropertyCondition = [
        PropertyFilter(
            path="StreetLight/visibility",
            operator="<",
            value="30",
            strategy=strategyFilters
        )
    ]
    devicePropertyConditionConditions = DeviceDataCondition(
        device_id="07b69d78-c716-4be6-9545-869920738397",
        filters=listFiltersDevicePropertyCondition
    )
    listConditionsConditionGroup = [
        RuleCondition(
            type="DEVICE_DATA",
            device_property_condition=devicePropertyConditionConditions
        )
    ]
    conditionGroupbody = ConditionGroup(
        conditions=listConditionsConditionGroup,
        logic="or",
        time_range=timeRangeConditionGroup
    )
    request.body = Rule(
        app_id="string",
        status="active",
        rule_type="DEVICE_LINKAGE",
        actions=listActionsbody,
        condition_group=conditionGroupbody,
        description="string",
        name="openLight"
    )
    response = client.create_rule(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

- 创建端测规则，属性上报触发命令下发。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
```

```
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRuleRequest()
        listDeviceIdsDeviceSide = [
            "3a9e52d9-3ebf-4985-89e9-6d2396748a2f"
        ]
        deviceSidebody = DeviceSide(
            device_ids=listDeviceIdsDeviceSide
        )
        cmdDeviceCommand = Cmd(
            command_name="SET_LIGHT_SWITCH",
            command_body="{\"value\":0}",
            service_id="Switch",
            buffer_timeout=0,
            mode="ACTIVE"
        )
        deviceCommandActions = ActionDeviceCommand(
            device_id="3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
            cmd=cmdDeviceCommand
        )
        listActionsbody = [
            RuleAction(
                type="DEVICE_CMD",
                device_command=deviceCommandActions
            )
        ]
        timeRangeConditionGroup = TimeRange(
            start_time="18:00",
            end_time="23:00",
            days_of_week="2,3,4,5,6"
        )
        strategyFilters = Strategy(
            trigger="reverse",
            event_valid_time=300
        )
        listFiltersDevicePropertyCondition = [
            PropertyFilter(
                path="StreetLight/visibility",
                operator="<",
                value="30",
                strategy=strategyFilters
            )
        ]
```

```
        devicePropertyConditionConditions = DeviceDataCondition(
            device_id="3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
            filters=listFiltersDevicePropertyCondition
        )
        listConditionsConditionGroup = [
            RuleCondition(
                type="DEVICE_DATA",
                device_property_condition=devicePropertyConditionConditions
            )
        ]
        conditionGroupbody = ConditionGroup(
            conditions=listConditionsConditionGroup,
            logic="or",
            time_range=timeRangeConditionGroup
        )
        request.body = Rule(
            device_side=deviceSidebody,
            app_id="string",
            status="active",
            rule_type="DEVICE_SIDE",
            actions=listActionsbody,
            condition_group=conditionGroupbody,
            description="string",
            name="openLight"
        )
        response = client.create_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建云端规则，设备下线触发告警。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
```

```
        // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
        WithRegion(region.NewRegion("cn-north-4", endpoint)).
        WithCredential(auth).
        Build())

    request := &model.CreateRuleRequest{}
    descriptionDeviceAlarm:= "设备下线"
    deviceAlarmActions := &model.ActionDeviceAlarm{
        Name: "设备下线",
        AlarmStatus: "fault",
        Severity: "warning",
        Description: &descriptionDeviceAlarm,
    }
    var listActionsbody = []model.RuleAction{
        {
            Type: "DEVICE_ALARM",
            DeviceAlarm: deviceAlarmActions,
        },
    }
    daysOfWeekTimeRange:= "2,3,4,5,6"
    timeRangeConditionGroup := &model.TimeRange{
        StartTime: "18:00",
        EndTime: "23:00",
        DaysOfWeek: &daysOfWeekTimeRange,
    }
    var listStatusListDeviceLinkageStatusCondition = []string{
        "OFFLINE",
    }
    deviceIdDeviceLinkageStatusCondition:= "07b69d78-c716-4be6-9545-869920738397"
    durationDeviceLinkageStatusCondition:= int32(0)
    deviceLinkageStatusConditionConditions := &model.DeviceLinkageStatusCondition{
        DeviceId: &deviceIdDeviceLinkageStatusCondition,
        StatusList: &listStatusListDeviceLinkageStatusCondition,
        Duration: &durationDeviceLinkageStatusCondition,
    }
    var listConditionsConditionGroup = []model.RuleCondition{
        {
            Type: "DEVICE_LINKAGE_STATUS",
            DeviceLinkageStatusCondition: deviceLinkageStatusConditionConditions,
        },
    }
    logicConditionGroup:= "or"
    conditionGroupbody := &model.ConditionGroup{
        Conditions: &listConditionsConditionGroup,
        Logic: &logicConditionGroup,
        TimeRange: timeRangeConditionGroup,
    }
    appIdRule:= "string"
    statusRule:= "active"
    descriptionRule:= "string"
    request.Body = &model.Rule{
        AppId: &appIdRule,
        Status: &statusRule,
        RuleType: "DEVICE_LINKAGE",
        Actions: listActionsbody,
        ConditionGroup: conditionGroupbody,
        Description: &descriptionRule,
        Name: "openLight",
    }
    response, err := client.CreateRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建云端规则，属性上报触发命令下发。

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRuleRequest{}
    var commandBodyCmd interface{} = "{\"value\":0}"
    bufferTimeoutCmd:= int32(0)
    modeCmd:= "ACTIVE"
    cmdDeviceCommand := &model.Cmd{
        CommandName: "SET_LIGHT_SWITCH",
        CommandBody: &commandBodyCmd,
        ServiceId: "Switch",
        BufferTimeout: &bufferTimeoutCmd,
        Mode: &modeCmd,
    }
    deviceIdDeviceCommand:= "3a9e52d9-3ebf-4985-89e9-6d2396748a2f"
    deviceCommandActions := &model.ActionDeviceCommand{
        DeviceId: &deviceIdDeviceCommand,
        Cmd: cmdDeviceCommand,
    }
    var listActionsbody = []model.RuleAction{
        {
            Type: "DEVICE_CMD",
            DeviceCommand: deviceCommandActions,
        },
    }
    daysOfWeekTimeRange:= "2,3,4,5,6"
    timeRangeConditionGroup := &model.TimeRange{
        StartTime: "18:00",
        EndTime: "23:00",
        DaysOfWeek: &daysOfWeekTimeRange,
    }
    triggerStrategy:= "reverse"
    eventValidTimeStrategy:= int32(300)
    strategyFilters := &model.Strategy{
        Trigger: &triggerStrategy,
        EventValidTime: &eventValidTimeStrategy,
```



```
    }
    valueFilters:= "30"
    var listFiltersDevicePropertyCondition = []model.PropertyFilter{
        {
            Path: "StreetLight/visibility",
            Operator: "<",
            Value: &valueFilters,
            Strategy: strategyFilters,
        },
    }
    deviceIdDevicePropertyCondition:= "07b69d78-c716-4be6-9545-869920738397"
    devicePropertyConditionConditions := &model.DeviceDataCondition{
        DeviceId: &deviceIdDevicePropertyCondition,
        Filters: &listFiltersDevicePropertyCondition,
    }
    var listConditionsConditionGroup = []model.RuleCondition{
        {
            Type: "DEVICE_DATA",
            DevicePropertyCondition: devicePropertyConditionConditions,
        },
    }
    logicConditionGroup:= "or"
    conditionGroupbody := &model.ConditionGroup{
        Conditions: &listConditionsConditionGroup,
        Logic: &logicConditionGroup,
        TimeRange: timeRangeConditionGroup,
    }
    appIdRule:= "string"
    statusRule:= "active"
    descriptionRule:= "string"
    request.Body = &model.Rule{
        AppId: &appIdRule,
        Status: &statusRule,
        RuleType: "DEVICE_LINKAGE",
        Actions: listActionsbody,
        ConditionGroup: conditionGroupbody,
        Description: &descriptionRule,
        Name: "openLight",
    }
    response, err := client.CreateRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建端测规则，属性上报触发命令下发。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"
```

```
    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRuleRequest{}
    var listDeviceIdsDeviceSide = []string{
        "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
    }
    deviceSidebody := &model.DeviceSide{
        DeviceIds: &listDeviceIdsDeviceSide,
    }
    var commandBodyCmd interface{} = "{\"value\":0}"
    bufferTimeoutCmd:= int32(0)
    modeCmd:= "ACTIVE"
    cmdDeviceCommand := &model.Cmd{
        CommandName: "SET_LIGHT_SWITCH",
        CommandBody: &commandBodyCmd,
        ServiceId: "Switch",
        BufferTimeout: &bufferTimeoutCmd,
        Mode: &modeCmd,
    }
    deviceIdDeviceCommand:= "3a9e52d9-3ebf-4985-89e9-6d2396748a2f"
    deviceCommandActions := &model.ActionDeviceCommand{
        DeviceId: &deviceIdDeviceCommand,
        Cmd: cmdDeviceCommand,
    }
    var listActionsbody = []model.RuleAction{
        {
            Type: "DEVICE_CMD",
            DeviceCommand: deviceCommandActions,
        },
    }
    daysOfWeekTimeRange:= "2,3,4,5,6"
    timeRangeConditionGroup := &model.TimeRange{
        StartTime: "18:00",
        EndTime: "23:00",
        DaysOfWeek: &daysOfWeekTimeRange,
    }
    triggerStrategy:= "reverse"
    eventValidTimeStrategy:= int32(300)
    strategyFilters := &model.Strategy{
        Trigger: &triggerStrategy,
        EventValidTime: &eventValidTimeStrategy,
    }
    valueFilters:= "30"
    var listFiltersDevicePropertyCondition = []model.PropertyFilter{
        {
            Path: "StreetLight/visibility",
            Operator: "<",
            Value: &valueFilters,
            Strategy: strategyFilters,
        },
    }
    deviceIdDevicePropertyCondition:= "3a9e52d9-3ebf-4985-89e9-6d2396748a2f"
    devicePropertyConditionConditions := &model.DeviceDataCondition{
        DeviceId: &deviceIdDevicePropertyCondition,
        Filters: &listFiltersDevicePropertyCondition,
```



```
    }
    var listConditionsConditionGroup = []model.RuleCondition{
        {
            Type: "DEVICE_DATA",
            DevicePropertyCondition: devicePropertyConditionConditions,
        },
    }
    logicConditionGroup:= "or"
    conditionGroupbody := &model.ConditionGroup{
        Conditions: &listConditionsConditionGroup,
        Logic: &logicConditionGroup,
        TimeRange: timeRangeConditionGroup,
    }
    appIdRule:= "string"
    statusRule:= "active"
    descriptionRule:= "string"
    request.Body = &model.Rule{
        DeviceSide: deviceSidebody,
        AppId: &appIdRule,
        Status: &statusRule,
        RuleType: "DEVICE_SIDE",
        Actions: listActionsbody,
        ConditionGroup: conditionGroupbody,
        Description: &descriptionRule,
        Name: "openLight",
    }
    response, err := client.CreateRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.10.2 查询规则列表

功能介绍

应用服务器可调用此接口查询物联网平台中设置的规则列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/rules

表 1-393 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID**。

表 1-394 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的规则列表，不携带该参数则会查询该用户下所有规则列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
rule_type	否	String	**参数说明**：规则类型。此参数为非必选参数，指定对应的规则类型结果进行返回，不携带该参数则会返回所有类型规则。 **取值范围**： • DEVICE_LINKAGE：云端联动规则。 • DEVICE_SIDE：端侧规则。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-395 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-396 响应 Body 参数

参数	参数类型	描述
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。
count	Long	满足查询条件的记录总数。
rules	Array of **RuleResponse** objects	规则信息列表。

表 1-397 RuleResponse

参数	参数类型	描述
rule_id	String	规则id。 最大长度：**128**
name	String	规则名称。 最小长度：**1** 最大长度：**128**

参数	参数类型	描述
description	String	规则的描述信息。 最大长度：**256**
condition_group	**ConditionGroup** object	规则的条件组，包含简单规则和复杂规则集合。
actions	Array of **RuleAction** objects	规则的动作列表，单个规则最多支持设置10个动作。
rule_type	String	规则的类型 • DEVICE_LINKAGE：云端联动规则。 • DEVICE_SIDE：端侧规则。
status	String	规则的状态，默认值：active。 • active：激活。 • inactive：未激活。
app_id	String	资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的规则归属到哪个资源空间下，否则创建的规则将会归属到**默认资源空间**下。
edge_node_ids	Array of strings	归属边缘侧节点设备ID列表。
last_update_time	String	规则最后更新时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。
device_side	**DeviceSide** object	**参数说明**：端侧执行下发设备信息，当规则类型为DEVICE_SIDE时，该参数必填。

表 1-398 ConditionGroup

参数	参数类型	描述
conditions	Array of **RuleCondition** objects	**参数说明**：规则的条件列表，单个规则最多支持设置10个条件。
logic	String	**参数说明**：规则条件列表中多个条件之间的逻辑关系，默认值：and。 **取值范围**： • and：逻辑且。 • or：逻辑或。
time_range	**TimeRange** object	**参数说明**：规则条件触发的有效时间段。

表 1-399 RuleCondition

参数	参数类型	描述
type	String	**参数说明**：规则条件的类型。 **取值范围**： • DEVICE_DATA：设备属性数据类型条件。 • SIMPLE_TIMER：简单定时类型条件。 • DAILY_TIMER：每日定时类型条件。 • DEVICE_LINKAGE_STATUS：设备状态类型条件。
device_property_condition	**DeviceDataCondition** object	**参数说明**：条件中设备数据类型的信息，当type为DEVICE_DATA时，为必选参数。
simple_timer_condition	**SimpleTimerType** object	**参数说明**：条件中简单定时类型的信息，当type为SIMPLE_TIMER时，为必选参数。
daily_timer_condition	**DailyTimerType** object	**参数说明**：条件中每日定时类型的信息，当type为DAILY_TIMER时，为必选参数。
device_linkage_status_condition	**DeviceLinkageStatusCondition** object	**参数说明**：条件中设备状态类型的信息，当规则类型为DEVICE_LINKAGE且type为DEVICE_LINKAGE_STATUS时，为必选参数。

表 1-400 DeviceDataCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备属性触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备属性触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
filters	Array of **PropertyFilter** objects	数据过滤条件。

表 1-401 PropertyFilter

参数	参数类型	描述
path	String	**参数说明**：设备属性的路径信息，格式：service_id/DataProperty，例如门磁状态为“DoorWindow/status”。 最大长度：**128**
operator	String	**参数说明**：数据比较的操作符。 **取值范围**：当前支持的操作符有：>，<，>=，<=，=，in:表示在指定值中匹配和between:表示数值区间。
value	String	**参数说明**：数据比较表达式的右值。与数据比较操作符between联用时，右值表示最小值和最大值，用逗号隔开，如“20,30”表示大于等于20小于30。 最大长度：**64**
in_values	Array of strings	**参数说明**：当operator为in时该字段必填，使用该字段传递比较表达式右值，上限为20个。 最大长度：**128**
strategy	**Strategy** object	**参数说明**：规则条件的处理策略，用于确定规则是否判断上次数据是否满足条件。当rule_type为DEVICE_LINKAGE时，该参数值不能为空。端侧执行不支持该字段。

表 1-402 Strategy

参数	参数类型	描述
trigger	String	**参数说明**：规则条件触发的判断策略，默认为pulse。 **取值范围**： • pulse：设备上报的数据满足条件则触发，不判断上一次上报的数据。 • reverse：设备上一次上报的数据不满足条件，本次上报的数据满足条件则触发。
event_valid_time	Integer	**参数说明**：设备数据的有效时间，单位为秒，设备数据的产生时间以上报数据中的eventTime为基准。 最小值：**-1**

表 1-403 SimpleTimerType

参数	参数类型	描述
start_time	String	**参数说明**：规则触发的开始时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。 最大长度：**128**
repeat_interval	Integer	**参数说明**：规则触发的重复时间间隔，单位为秒。 最小值：**1** 最大值：**31536000**
repeat_count	Integer	**参数说明**：规则触发的重复次数。 最小值：**1** 最大值：**9999**

表 1-404 DailyTimerType

参数	参数类型	描述
time	String	**参数说明**：规则触发的时间，格式：HH:MM。 最大长度：**128**
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。 最大长度：**128**

表 1-405 DeviceLinkageStatusCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备状态触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备状态触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**

参数	参数类型	描述
status_list	Array of strings	**参数说明**：状态列表，设备状态条件携带该参数。 **取值范围**： • ONLINE：设备上线 • OFFLINE：设备下线 最大长度：**128**
duration	Integer	**持续时长**：设备状态持续时长，取值范围:0-60(分钟)。 最小值：**0** 最大值：**60** 缺省值：**0**

表 1-406 TimeRange

参数	参数类型	描述
start_time	String	**参数说明**：规则条件触发的开始时间，格式：HH:mm。
end_time	String	**参数说明**：规则条件触发的结束时间，格式：HH:mm。若结束时间与开始时间一致，则时间为全天。
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。星期列表中的日期为开始时间的日期。

表 1-407 RuleAction

参数	参数类型	描述
type	String	**参数说明**：规则动作的类型，端侧执行只支持下发设备命令消息类型。 **取值范围**： • DEVICE_CMD：下发设备命令消息类型。 • SMN_FORWARDING：发送SMN消息类型。 • DEVICE_ALARM：上报设备告警消息类型。当选择该类型时，condition中必须有DEVICE_DATA条件类型。该类型动作只能唯一。
device_command	**ActionDeviceCommand** object	下发设备命令消息内容。当type为DEVICE_CMD时，必填。

参数	参数类型	描述
smn_forwarding	**ActionSmnForwarding** object	发送给SMN消息结构。当规则类型为DEVICE_LINKAGE且type为SMN_FORWARDING时，必填。
device_alarm	**ActionDeviceAlarm** object	上报设备告警消息内容。当规则类型为DEVICE_LINKAGE且type为DEVICE_ALARM时，必填。

表 1-408 ActionDeviceCommand

参数	参数类型	描述
device_id	String	**参数说明**：下发命令的设备ID。 • 当创建设备数据规则时，若device_id为空，则命令下发给触发条件的设备。 • 当创建定时规则时，不允许为空。 **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。
cmd	**CMD** object	**参数说明**：下发的命令信息。

表 1-409 CMD

参数	参数类型	描述
command_name	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。 最大长度：**128**

参数	参数类型	描述
command_body	Object	**参数说明**：设备命令参数，Json格式。 使用LWM2M协议设备命令示例：{"value":"1"}，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。 使用MQTT协议设备命令示例：{"header": {"mode": "ACK","from": "/users/testUser","method": "SET_TEMPERATURE_READ_PERIOD","to":"/devices/{device_id}/services/{service_id}"},"body": {"value" : "1"}}。 ● mode：必选，设备收到命令后是否需要回复确认消息，默认为ACK模式。ACK表示需要回复确认消息，NOACK表示不需要回复确认消息，其它值无效。 ● from：可选，命令发送方的地址。App发起请求时格式为/users/{userId} ，应用服务器发起请求时格式为/{serviceName}，物联网平台发起请求时格式为/cloud/{serviceName}。 ● to：可选，命令接收方的地址，格式为/devices/{device_id}/services/{service_id}。 ● method：可选，产品模型中定义的命令名称。 ● body：可选，命令的消息体，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。具体格式需要应用和设备约定。
service_id	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。 最大长度：**64**
buffer_timeout	Integer	**参数说明**：设备命令的缓存时间，单位为秒，表示物联网平台在把命令下发给设备前缓存命令的有效时间，超过这个时间后命令将不再下发，默认值为172800s（48小时）。 如果buffer_timeout设置为0，则无论物联网平台上设置的命令下发模式是什么，该命令都会立即下发给设备。 最小值：**0** 最大值：**31536000** 缺省值：**172800**

参数	参数类型	描述
response_timeout	Integer	**参数说明**：命令响应的有效时间，单位为秒，表示设备接收到命令后，在response_timeout时间内响应有效，超过这个时间未收到命令的响应，则认为命令响应超时，默认值为1800s。 最小值：**1** 最大值：**31536000** 缺省值：**1800**
mode	String	**参数说明**：设备命令的下发模式，仅当buffer_timeout的值大于0时有效。 ● ACTIVE：主动模式，物联网平台主动将命令下发给设备。 ● PASSIVE：被动模式，物联网平台创建设备命令后，会直接缓存命令。等到设备再次上线或者上报上一条命令的执行结果后才下发命令。

表 1-410 ActionSmnForwarding

参数	参数类型	描述
region_name	String	**参数说明**：SMN服务对应的region区域。 最大长度：**256**
project_id	String	**参数说明**：SMN服务对应的projectId信息。
theme_name	String	**参数说明**：SMN服务对应的主题名称。 最大长度：**256**
topic_urn	String	**参数说明**：SMN服务对应的topic的主题URN。 最大长度：**256**
message_content	String	**参数说明**：短信或邮件的内容。 最大长度：**1024**
message_template_name	String	**参数说明**：SMN服务对应的模板名称。
message_title	String	**参数说明**：短信或邮件的主题。最大长度支持UTF-8编码后的521个字节。

表 1-411 ActionDeviceAlarm

参数	参数类型	描述
name	String	**参数说明**：告警名称。 最大长度：**256**

参数	参数类型	描述
alarm_status	String	**参数说明**：告警状态。 **取值范围**： ● fault：上报告警。 ● recovery：恢复告警。
severity	String	**参数说明**：告警级别。 **取值范围**：warning（警告）、minor（一般）、major（严重）和critical（致命）。
dimension	String	**参数说明**：告警维度，与告警名称和告警级别组合起来共同标识一条告警，默认不携带该字段为用户维度告警，支持设备维度和资源空间维度告警。 **取值范围**： ● device：设备维度。 ● app：资源空间维度。
description	String	**参数说明**：告警的描述信息。 最大长度：**256**

表 1-412 DeviceSide

参数	参数类型	描述
device_ids	Array of strings	**参数说明**：端侧执行下发的目标设备ID列表。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。 最大长度：**128**

请求示例

列表查询指定资源空间的联动规则。

```
GET https://{endpoint}/v5/iot/{project_id}/rules?
app_id={app_id}&limit={limit}&marker={marker}&offset={offset}
```

响应示例

状态码： 200

OK

```
{
  "marker" : "6336a282cda07a01f7ac5d11",
  "count" : 1,
  "rules" : [ {
    "rule_id" : "5eb3628d017d9105d0cf9aec",
    "name" : "openLight",
    "condition_group" : {
      "conditions" : [ {
        "type" : "DEVICE_DATA",
        "device_property_condition" : {
          "device_id" : "07b69d78-c716-4be6-9545-869920738397",
```



```
        "filters" : [ {
          "path" : "StreetLight/visibility",
          "operator" : "<",
          "value" : "30",
          "strategy" : {
            "trigger" : "reverse",
            "event_valid_time" : 300
          }
        } ]
      }
    } ],
    "logic" : "and"
  },
  "actions" : [ {
    "type" : "DEVICE_CMD",
    "device_command" : {
      "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
      "cmd" : {
        "command_name" : "SET_LIGHT_SWITCH",
        "command_body" : {
          "value" : 0
        },
        "service_id" : "Switch",
        "buffer_timeout" : 0,
        "response_timeout" : 1800
      }
    }
  } ],
  "rule_type" : "DEVICE_LINKAGE",
  "status" : "active",
  "app_id" : "9562bf8541e44361b6ae3a7e9fbe1144",
  "last_update_time" : "20221017T023727Z"
 } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListRulesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
```

```
            // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
            .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
            .withAk(ak)
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        ListRulesRequest request = new ListRulesRequest();
        request.withAppId("<app_id>");
        request.withRuleType("<rule_type>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListRulesResponse response = client.listRules(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListRulesRequest()
```



```
        request.app_id = "<app_id>"
        request.rule_type = "<rule_type>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_rules(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListRulesRequest{}
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    ruleTypeRequest:= "<rule_type>"
    request.RuleType = &ruleTypeRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListRules(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```



```
  }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.10.3 修改规则

功能介绍

应用服务器可调用此接口修改物联网平台中指定规则的配置。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/rules/{rule_id}

表 1-413 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID**。

参数	是否必选	参数类型	描述
rule_id	是	String	**参数说明**：规则ID，用于唯一标识一条规则，在创建规则时由物联网平台分配获得。 **取值范围**：长度不超过32，只允许字母、数字的组合。

请求参数

表 1-414 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-415 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	String	**参数说明**：规则名称。 最小长度：**1** 最大长度：**128**
description	否	String	**参数说明**：规则的描述信息。 最大长度：**256**
condition_group	是	**ConditionGroup** object	**参数说明**：规则的条件组，包含简单规则和复杂规则集合。
actions	是	Array of **RuleAction** objects	**参数说明**：规则的动作列表，单个规则最多支持设置10个动作。

参数	是否必选	参数类型	描述
rule_type	是	String	**参数说明**：规则的类型。 **取值范围**： • DEVICE_LINKAGE：云端联动规则。 • DEVICE_SIDE：端侧规则。
status	否	String	**参数说明**：规则的状态，默认值：active。 **取值范围**： • active：激活。 • inactive：未激活。
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的规则归属到哪个资源空间下，否则创建的规则将会归属到**默认资源空间**下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
device_side	否	**DeviceSide** object	**参数说明**：端侧执行下发设备信息，当规则类型为DEVICE_SIDE时，该参数必填。

表 1-416 ConditionGroup

参数	是否必选	参数类型	描述
conditions	否	Array of **RuleCondition** objects	**参数说明**：规则的条件列表，单个规则最多支持设置10个条件。
logic	否	String	**参数说明**：规则条件列表中多个条件之间的逻辑关系，默认值：and。 **取值范围**： • and：逻辑且。 • or：逻辑或。
time_range	否	**TimeRange** object	**参数说明**：规则条件触发的有效时间段。

表 1-417 RuleCondition

参数	是否必选	参数类型	描述
type	是	String	**参数说明**：规则条件的类型。 **取值范围**： • DEVICE_DATA：设备属性数据类型条件。 • SIMPLE_TIMER：简单定时类型条件。 • DAILY_TIMER：每日定时类型条件。 • DEVICE_LINKAGE_STATUS：设备状态类型条件。
device_property_condition	否	**DeviceDataCondition** object	**参数说明**：条件中设备数据类型的信息，当type为DEVICE_DATA时，为必选参数。
simple_timer_condition	否	**SimpleTimerType** object	**参数说明**：条件中简单定时类型的信息，当type为SIMPLE_TIMER时，为必选参数。
daily_timer_condition	否	**DailyTimerType** object	**参数说明**：条件中每日定时类型的信息，当type为DAILY_TIMER时，为必选参数。
device_linkage_status_condition	否	**DeviceLinkageStatusCondition** object	**参数说明**：条件中设备状态类型的信息，当规则类型为DEVICE_LINKAGE且type为DEVICE_LINKAGE_STATUS时，为必选参数。

表 1-418 DeviceDataCondition

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：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。存在该参数时设备属性触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。

参数	是否必选	参数类型	描述
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备属性触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
filters	否	Array of **PropertyFilter** objects	数据过滤条件。

表 1-419 PropertyFilter

参数	是否必选	参数类型	描述
path	是	String	**参数说明**：设备属性的路径信息，格式：service_id/DataProperty，例如门磁状态为“DoorWindow/status”。 最大长度：**128**
operator	是	String	**参数说明**：数据比较的操作符。 **取值范围**：当前支持的操作符有：>，<，>=，<=，=，in:表示在指定值中匹配和between:表示数值区间。
value	否	String	**参数说明**：数据比较表达式的右值。与数据比较操作符between联用时，右值表示最小值和最大值，用逗号隔开，如“20,30”表示大于等于20小于30。 最大长度：**64**
in_values	否	Array of strings	**参数说明**：当operator为in时该字段必填，使用该字段传递比较表达式右值，上限为20个。 最大长度：**128**
strategy	否	**Strategy** object	**参数说明**：规则条件的处理策略，用于确定规则是否判断上次数据是否满足条件。当rule_type为DEVICE_LINKAGE时，该参数值不能为空。端侧执行不支持该字段。

表 1-420 Strategy

参数	是否必选	参数类型	描述
trigger	否	String	**参数说明**：规则条件触发的判断策略，默认为pulse。 **取值范围**： • pulse：设备上报的数据满足条件则触发，不判断上一次上报的数据。 • reverse：设备上一次上报的数据不满足条件，本次上报的数据满足条件则触发。
event_valid_time	否	Integer	**参数说明**：设备数据的有效时间，单位为秒，设备数据的产生时间以上报数据中的eventTime为基准。 最小值：**-1**

表 1-421 SimpleTimerType

参数	是否必选	参数类型	描述
start_time	是	String	**参数说明**：规则触发的开始时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。 最大长度：**128**
repeat_interval	是	Integer	**参数说明**：规则触发的重复时间间隔，单位为秒。 最小值：**1** 最大值：**31536000**
repeat_count	是	Integer	**参数说明**：规则触发的重复次数。 最小值：**1** 最大值：**9999**

表 1-422 DailyTimerType

参数	是否必选	参数类型	描述
time	是	String	**参数说明**：规则触发的时间，格式：HH:MM。 最大长度：**128**

参数	是否必选	参数类型	描述
days_of_week	否	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。 最大长度：**128**

表 1-423 DeviceLinkageStatusCondition

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：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。存在该参数时设备状态触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备状态触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
status_list	否	Array of strings	**参数说明**：状态列表，设备状态条件携带该参数。 **取值范围**： • ONLINE：设备上线 • OFFLINE：设备下线 最大长度：**128**
duration	否	Integer	**持续时长**：设备状态持续时长，取值范围: 0-60(分钟)。 最小值：**0** 最大值：**60** 缺省值：**0**

表 1-424 TimeRange

参数	是否必选	参数类型	描述
start_time	是	String	**参数说明**：规则条件触发的开始时间，格式：HH:mm。
end_time	是	String	**参数说明**：规则条件触发的结束时间，格式：HH:mm。若结束时间与开始时间一致，则时间为全天。
days_of_week	否	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。星期列表中的日期为开始时间的日期。

表 1-425 RuleAction

参数	是否必选	参数类型	描述
type	是	String	**参数说明**：规则动作的类型，端侧执行只支持下发设备命令消息类型。 **取值范围**： • DEVICE_CMD：下发设备命令消息类型。 • SMN_FORWARDING：发送SMN消息类型。 • DEVICE_ALARM：上报设备告警消息类型。当选择该类型时，condition中必须有DEVICE_DATA条件类型。该类型动作只能唯一。
device_command	否	**ActionDeviceCommand** object	下发设备命令消息内容。当type为DEVICE_CMD时，必填。
smn_forwarding	否	**ActionSmnForwarding** object	发送给SMN消息结构。当规则类型为DEVICE_LINKAGE且type为SMN_FORWARDING时，必填。
device_alarm	否	**ActionDeviceAlarm** object	上报设备告警消息内容。当规则类型为DEVICE_LINKAGE且type为DEVICE_ALARM时，必填。

表 1-426 ActionDeviceCommand

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：下发命令的设备ID。 • 当创建设备数据规则时，若device_id为空，则命令下发给触发条件的设备。 • 当创建定时规则时，不允许为空。 **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。
cmd	是	**CMD** object	**参数说明**：下发的命令信息。

表 1-427 CMD

参数	是否必选	参数类型	描述
command_name	是	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。 最大长度：**128**

参数	是否必选	参数类型	描述
command_body	是	Object	**参数说明**：设备命令参数，Json格式。 使用LWM2M协议设备命令示例：{"value":"1"}，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。 使用MQTT协议设备命令示例：{"header":{"mode": "ACK","from": "/users/testUser","method":"SET_TEMPERATURE_READ_PERIOD","to":"/devices/{device_id}/services/{service_id}"},"body":{"value" : "1"}}。 ● mode：必选，设备收到命令后是否需要回复确认消息，默认为ACK模式。ACK表示需要回复确认消息，NOACK表示不需要回复确认消息，其它值无效。 ● from：可选，命令发送方的地址。App发起请求时格式为/users/{userId}，应用服务器发起请求时格式为/{serviceName}，物联网平台发起请求时格式为/cloud/{serviceName}。 ● to：可选，命令接收方的地址，格式为/devices/{device_id}/services/{service_id}。 ● method：可选，产品模型中定义的命令名称。 ● body：可选，命令的消息体，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。具体格式需要应用和设备约定。
service_id	是	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。 最大长度：**64**

参数	是否必选	参数类型	描述
buffer_timeout	否	Integer	**参数说明**：设备命令的缓存时间，单位为秒，表示物联网平台在把命令下发给设备前缓存命令的有效时间，超过这个时间后命令将不再下发，默认值为172800s（48小时）。 如果buffer_timeout设置为0，则无论物联网平台上设置的命令下发模式是什么，该命令都会立即下发给设备。 最小值：**0** 最大值：**31536000** 缺省值：**172800**
response_timeout	否	Integer	**参数说明**：命令响应的有效时间，单位为秒，表示设备接收到命令后，在response_timeout时间内响应有效，超过这个时间未收到命令的响应，则认为命令响应超时，默认值为1800s。 最小值：**1** 最大值：**31536000** 缺省值：**1800**
mode	否	String	**参数说明**：设备命令的下发模式，仅当buffer_timeout的值大于0时有效。 ● ACTIVE：主动模式，物联网平台主动将命令下发给设备。 ● PASSIVE：被动模式，物联网平台创建设备命令后，会直接缓存命令。等到设备再次上线或者上报上一条命令的执行结果后才下发命令。

表 1-428 ActionSmnForwarding

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：SMN服务对应的region区域。 最大长度：**256**
project_id	是	String	**参数说明**：SMN服务对应的projectId信息。

参数	是否必选	参数类型	描述
theme_name	是	String	**参数说明**：SMN服务对应的主题名称。 最大长度：**256**
topic_urn	是	String	**参数说明**：SMN服务对应的topic的主题URN。 最大长度：**256**
message_content	否	String	**参数说明**：短信或邮件的内容。 最大长度：**1024**
message_template_name	否	String	**参数说明**：SMN服务对应的模板名称。
message_title	是	String	**参数说明**：短信或邮件的主题。最大长度支持UTF-8编码后的521个字节。

表 1-429 ActionDeviceAlarm

参数	是否必选	参数类型	描述
name	是	String	**参数说明**：告警名称。 最大长度：**256**
alarm_status	是	String	**参数说明**：告警状态。 **取值范围**： ● fault：上报告警。 ● recovery：恢复告警。
severity	是	String	**参数说明**：告警级别。 **取值范围**：warning（警告）、minor（一般）、major（严重）和critical（致命）。
dimension	否	String	**参数说明**：告警维度，与告警名称和告警级别组合起来共同标识一条告警，默认不携带该字段为用户维度告警，支持设备维度和资源空间维度告警。 **取值范围**： ● device：设备维度。 ● app：资源空间维度。
description	否	String	**参数说明**：告警的描述信息。 最大长度：**256**

表 1-430 DeviceSide

参数	是否必选	参数类型	描述
device_ids	否	Array of strings	**参数说明**：端侧执行下发的目标设备ID列表。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。 最大长度：**128**

响应参数

状态码： 200

表 1-431 响应 Body 参数

参数	参数类型	描述
rule_id	String	规则id。 最大长度：**128**
name	String	规则名称。 最小长度：**1** 最大长度：**128**
description	String	规则的描述信息。 最大长度：**256**
condition_group	**ConditionGroup** object	规则的条件组，包含简单规则和复杂规则集合。
actions	Array of **RuleAction** objects	规则的动作列表，单个规则最多支持设置10个动作。
rule_type	String	规则的类型 • DEVICE_LINKAGE：云端联动规则。 • DEVICE_SIDE：端侧规则。
status	String	规则的状态，默认值：active。 • active：激活。 • inactive：未激活。
app_id	String	资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的规则归属到哪个资源空间下，否则创建的规则将会归属到**默认资源空间**下。
edge_node_ids	Array of strings	归属边缘侧节点设备ID列表。

参数	参数类型	描述
last_update_time	String	规则最后更新时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。
device_side	**DeviceSide** object	**参数说明**：端侧执行下发设备信息，当规则类型为DEVICE_SIDE时，该参数必填。

表 1-432 ConditionGroup

参数	参数类型	描述
conditions	Array of **RuleCondition** objects	**参数说明**：规则的条件列表，单个规则最多支持设置10个条件。
logic	String	**参数说明**：规则条件列表中多个条件之间的逻辑关系，默认值：and。 **取值范围**： • and：逻辑且。 • or：逻辑或。
time_range	**TimeRange** object	**参数说明**：规则条件触发的有效时间段。

表 1-433 RuleCondition

参数	参数类型	描述
type	String	**参数说明**：规则条件的类型。 **取值范围**： • DEVICE_DATA：设备属性数据类型条件。 • SIMPLE_TIMER：简单定时类型条件。 • DAILY_TIMER：每日定时类型条件。 • DEVICE_LINKAGE_STATUS：设备状态类型条件。
device_property_condition	**DeviceDataCondition** object	**参数说明**：条件中设备数据类型的信息，当type为DEVICE_DATA时，为必选参数。
simple_timer_condition	**SimpleTimerType** object	**参数说明**：条件中简单定时类型的信息，当type为SIMPLE_TIMER时，为必选参数。
daily_timer_condition	**DailyTimerType** object	**参数说明**：条件中每日定时类型的信息，当type为DAILY_TIMER时，为必选参数。
device_linkage_status_condition	**DeviceLinkageStatusCondition** object	**参数说明**：条件中设备状态类型的信息，当规则类型为DEVICE_LINKAGE且type为DEVICE_LINKAGE_STATUS时，为必选参数。

表 1-434 DeviceDataCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备属性触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备属性触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
filters	Array of **PropertyFilter** objects	数据过滤条件。

表 1-435 PropertyFilter

参数	参数类型	描述
path	String	**参数说明**：设备属性的路径信息，格式：service_id/DataProperty，例如门磁状态为“DoorWindow/status”。 最大长度：**128**
operator	String	**参数说明**：数据比较的操作符。 **取值范围**：当前支持的操作符有：>，<，>=，<=，=，in:表示在指定值中匹配和between:表示数值区间。
value	String	**参数说明**：数据比较表达式的右值。与数据比较操作符between联用时，右值表示最小值和最大值，用逗号隔开，如“20,30”表示大于等于20小于30。 最大长度：**64**
in_values	Array of strings	**参数说明**：当operator为in时该字段必填，使用该字段传递比较表达式右值，上限为20个。 最大长度：**128**
strategy	**Strategy** object	**参数说明**：规则条件的处理策略，用于确定规则是否判断上次数据是否满足条件。当rule_type为DEVICE_LINKAGE时，该参数值不能为空。端侧执行不支持该字段。

表 1-436 Strategy

参数	参数类型	描述
trigger	String	**参数说明**：规则条件触发的判断策略，默认为pulse。 **取值范围**： ● pulse：设备上报的数据满足条件则触发，不判断上一次上报的数据。 ● reverse：设备上一次上报的数据不满足条件，本次上报的数据满足条件则触发。
event_valid_time	Integer	**参数说明**：设备数据的有效时间，单位为秒，设备数据的产生时间以上报数据中的eventTime为基准。 最小值：**-1**

表 1-437 SimpleTimerType

参数	参数类型	描述
start_time	String	**参数说明**：规则触发的开始时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。 最大长度：**128**
repeat_interval	Integer	**参数说明**：规则触发的重复时间间隔，单位为秒。 最小值：**1** 最大值：**31536000**
repeat_count	Integer	**参数说明**：规则触发的重复次数。 最小值：**1** 最大值：**9999**

表 1-438 DailyTimerType

参数	参数类型	描述
time	String	**参数说明**：规则触发的时间，格式：HH:MM。 最大长度：**128**
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。 最大长度：**128**

表 1-439 DeviceLinkageStatusCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备状态触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备状态触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
status_list	Array of strings	**参数说明**：状态列表，设备状态条件携带该参数。 **取值范围**： ● ONLINE：设备上线 ● OFFLINE：设备下线 最大长度：**128**
duration	Integer	**持续时长**：设备状态持续时长，取值范围:0-60(分钟)。 最小值：**0** 最大值：**60** 缺省值：**0**

表 1-440 TimeRange

参数	参数类型	描述
start_time	String	**参数说明**：规则条件触发的开始时间，格式：HH:mm。
end_time	String	**参数说明**：规则条件触发的结束时间，格式：HH:mm。若结束时间与开始时间一致，则时间为全天。
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。星期列表中的日期为开始时间的日期。

表 1-441 RuleAction

参数	参数类型	描述
type	String	**参数说明**：规则动作的类型，端侧执行只支持下发设备命令消息类型。 **取值范围**： • DEVICE_CMD：下发设备命令消息类型。 • SMN_FORWARDING：发送SMN消息类型。 • DEVICE_ALARM：上报设备告警消息类型。当选择该类型时，condition中必须有DEVICE_DATA条件类型。该类型动作只能唯一。
device_command	**ActionDeviceCommand** object	下发设备命令消息内容。当type为DEVICE_CMD时，必填。
smn_forwarding	**ActionSmnForwarding** object	发送给SMN消息结构。当规则类型为DEVICE_LINKAGE且type为SMN_FORWARDING时，必填。
device_alarm	**ActionDeviceAlarm** object	上报设备告警消息内容。当规则类型为DEVICE_LINKAGE且type为DEVICE_ALARM时，必填。

表 1-442 ActionDeviceCommand

参数	参数类型	描述
device_id	String	**参数说明**：下发命令的设备ID。 • 当创建设备数据规则时，若device_id为空，则命令下发给触发条件的设备。 • 当创建定时规则时，不允许为空。 **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。
cmd	**CMD** object	**参数说明**：下发的命令信息。

表 1-443 CMD

参数	参数类型	描述
command_name	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。 最大长度：**128**

参数	参数类型	描述
command_body	Object	**参数说明**：设备命令参数，Json格式。 使用LWM2M协议设备命令示例：{"value":"1"}，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。 使用MQTT协议设备命令示例：{"header": {"mode": "ACK","from": "/users/testUser","method": "SET_TEMPERATURE_READ_PERIOD","to":"/devices/{device_id}/services/{service_id}"},"body": {"value" : "1"}}。 ● mode：必选，设备收到命令后是否需要回复确认消息，默认为ACK模式。ACK表示需要回复确认消息，NOACK表示不需要回复确认消息，其它值无效。 ● from：可选，命令发送方的地址。App发起请求时格式为/users/{userId}，应用服务器发起请求时格式为/{serviceName}，物联网平台发起请求时格式为/cloud/{serviceName}。 ● to：可选，命令接收方的地址，格式为/devices/{device_id}/services/{service_id}。 ● method：可选，产品模型中定义的命令名称。 ● body：可选，命令的消息体，里面是一个个键值对，每个键都是产品模型中命令的参数名（paraName）。具体格式需要应用和设备约定。
service_id	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。 最大长度：**64**
buffer_timeout	Integer	**参数说明**：设备命令的缓存时间，单位为秒，表示物联网平台在把命令下发给设备前缓存命令的有效时间，超过这个时间后命令将不再下发，默认值为172800s（48小时）。 如果buffer_timeout设置为0，则无论物联网平台上设置的命令下发模式是什么，该命令都会立即下发给设备。 最小值：**0** 最大值：**31536000** 缺省值：**172800**

参数	参数类型	描述
response_timeout	Integer	**参数说明**：命令响应的有效时间，单位为秒，表示设备接收到命令后，在response_timeout时间内响应有效，超过这个时间未收到命令的响应，则认为命令响应超时，默认值为1800s。 最小值：**1** 最大值：**31536000** 缺省值：**1800**
mode	String	**参数说明**：设备命令的下发模式，仅当buffer_timeout的值大于0时有效。 ● ACTIVE：主动模式，物联网平台主动将命令下发给设备。 ● PASSIVE：被动模式，物联网平台创建设备命令后，会直接缓存命令。等到设备再次上线或者上报上一条命令的执行结果后才下发命令。

表 1-444 ActionSmnForwarding

参数	参数类型	描述
region_name	String	**参数说明**：SMN服务对应的region区域。 最大长度：**256**
project_id	String	**参数说明**：SMN服务对应的projectId信息。
theme_name	String	**参数说明**：SMN服务对应的主题名称。 最大长度：**256**
topic_urn	String	**参数说明**：SMN服务对应的topic的主题URN。 最大长度：**256**
message_content	String	**参数说明**：短信或邮件的内容。 最大长度：**1024**
message_template_name	String	**参数说明**：SMN服务对应的模板名称。
message_title	String	**参数说明**：短信或邮件的主题。最大长度支持UTF-8编码后的521个字节。

表 1-445 ActionDeviceAlarm

参数	参数类型	描述
name	String	**参数说明**：告警名称。 最大长度：**256**

参数	参数类型	描述
alarm_status	String	**参数说明**：告警状态。 **取值范围**： ● fault：上报告警。 ● recovery：恢复告警。
severity	String	**参数说明**：告警级别。 **取值范围**：warning（警告）、minor（一般）、major（严重）和critical（致命）。
dimension	String	**参数说明**：告警维度，与告警名称和告警级别组合起来共同标识一条告警，默认不携带该字段为用户维度告警，支持设备维度和资源空间维度告警。 **取值范围**： ● device：设备维度。 ● app：资源空间维度。
description	String	**参数说明**：告警的描述信息。 最大长度：**256**

表 1-446 DeviceSide

参数	参数类型	描述
device_ids	Array of strings	**参数说明**：端侧执行下发的目标设备ID列表。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。 最大长度：**128**

请求示例

修改联动规则，在周二的18点至24点运行规则，当满足visibility小于30或者到19点或者到达指定时触发告警设备属异常，并下发设备命令。

```
PUT https://{endpoint}/v5/iot/{project_id}/rules/{rule_id}

{
  "name" : "openLight",
  "description" : "string",
  "condition_group" : {
    "time_range" : {
      "days_of_week" : 2,
      "start_time" : "18:00",
      "end_time" : "23:00"
    },
    "logic" : "or",
    "conditions" : [ {
      "device_property_condition" : {
        "device_id" : "07b69d78-c716-4be6-9545-869920738397",
        "product_id" : "074abacf-cdb1-4f52-8c88-5864e50d332c",
        "filters" : [ {
          "path" : "StreetLight/visibility",
          "strategy" : {
            "event_valid_time" : 300,
```



```
          "trigger" : "reverse"
        },
        "value" : "30",
        "operator" : "<"
      } ]
    },
    "daily_timer_condition" : {
      "days_of_week" : 2,
      "time" : "19:00"
    },
    "type" : "DEVICE_DATA",
    "simple_timer_condition" : {
      "start_time" : "20190122T141500Z",
      "repeat_interval" : 1,
      "repeat_count" : 1
    }
  } ]
},
"actions" : [ {
  "device_alarm" : {
    "severity" : "warning",
    "alarm_status" : "fault",
    "name" : "设备属性异常",
    "description" : "****设备属性异常。"
  },
  "type" : "DEVICE_CMD",
  "smn_forwarding" : {
    "message_content" : "message_content",
    "project_id" : "project_id",
    "message_title" : "message_title",
    "theme_name" : "theme_name",
    "region_name" : "region_name",
    "topic_urn" : "topic_urn"
  },
  "device_command" : {
    "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
    "cmd" : {
      "buffer_timeout" : 0,
      "mode" : "string",
      "response_timeout" : 1800,
      "command_body" : {
        "value" : 0
      },
      "service_id" : "Switch",
      "command_name" : "SET_LIGHT_SWITCH"
    }
  }
} ],
"rule_type" : "DEVICE_LINKAGE",
"status" : "string",
"app_id" : "string"
}
```

响应示例

状态码： 200

OK

```
{
  "rule_id" : "5eb3628d017d9105d0cf9aec",
  "name" : "openLight",
  "condition_group" : {
    "conditions" : [ {
      "type" : "DEVICE_DATA",
      "device_property_condition" : {
        "device_id" : "07b69d78-c716-4be6-9545-869920738397",
        "filters" : [ {
          "path" : "StreetLight/visibility",
```



```
        "operator" : "<",
        "value" : "30",
        "strategy" : {
          "trigger" : "reverse",
          "event_valid_time" : 300
        }
      } ]
    }
  } ],
  "logic" : "and"
 },
 "actions" : [ {
  "type" : "DEVICE_CMD",
  "device_command" : {
    "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
    "cmd" : {
      "command_name" : "SET_LIGHT_SWITCH",
      "command_body" : {
        "value" : 0
      },
      "service_id" : "Switch",
      "buffer_timeout" : 0,
      "response_timeout" : 1800
    }
  }
 } ],
 "rule_type" : "DEVICE_LINKAGE",
 "status" : "active",
 "app_id" : "9562bf8541e44361b6ae3a7e9fbe1144",
 "last_update_time" : "20221017T025425Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改联动规则，在周二的18点至24点运行规则，当满足visibility小于30或者到19点或者到达指定时触发告警设备属异常，并下发设备命令。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
```

```
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateRuleRequest request = new UpdateRuleRequest();
        Rule body = new Rule();
        ActionDeviceAlarm deviceAlarmActions = new ActionDeviceAlarm();
        deviceAlarmActions.withName("设备属性异常")
            .withAlarmStatus("fault")
            .withSeverity("warning")
            .withDescription("****设备属性异常。");
        ActionSmnForwarding smnForwardingActions = new ActionSmnForwarding();
        smnForwardingActions.withRegionName("region_name")
            .withProjectId("project_id")
            .withThemeName("theme_name")
            .withTopicUrn("topic_urn")
            .withMessageContent("message_content")
            .withMessageTitle("message_title");
        Cmd cmdDeviceCommand = new Cmd();
        cmdDeviceCommand.withCommandName("SET_LIGHT_SWITCH")
            .withCommandBody("{\"value\":0}")
            .withServiceId("Switch")
            .withBufferTimeout(0)
            .withResponseTimeout(1800)
            .withMode("string");
        ActionDeviceCommand deviceCommandActions = new ActionDeviceCommand();
        deviceCommandActions.withDeviceId("3a9e52d9-3ebf-4985-89e9-6d2396748a2f")
            .withCmd(cmdDeviceCommand);
        List<RuleAction> listbodyActions = new ArrayList<>();
        listbodyActions.add(
            new RuleAction()
                .withType("DEVICE_CMD")
                .withDeviceCommand(deviceCommandActions)
                .withSmnForwarding(smnForwardingActions)
                .withDeviceAlarm(deviceAlarmActions)
        );
        TimeRange timeRangeConditionGroup = new TimeRange();
        timeRangeConditionGroup.withStartTime("18:00")
            .withEndTime("23:00")
            .withDaysOfWeek("2");
        DailyTimerType dailyTimerConditionConditions = new DailyTimerType();
        dailyTimerConditionConditions.withTime("19:00")
            .withDaysOfWeek("2");
        SimpleTimerType simpleTimerConditionConditions = new SimpleTimerType();
        simpleTimerConditionConditions.withStartTime("20190122T141500Z")
            .withRepeatInterval(1)
            .withRepeatCount(1);
        Strategy strategyFilters = new Strategy();
        strategyFilters.withTrigger("reverse")
            .withEventValidTime(300);
        List<PropertyFilter> listDevicePropertyConditionFilters = new ArrayList<>();
        listDevicePropertyConditionFilters.add(
            new PropertyFilter()
                .withPath("StreetLight/visibility")
                .withOperator("<")
                .withValue("30")
                .withStrategy(strategyFilters)
        );
```



```
        DeviceDataCondition devicePropertyConditionConditions = new DeviceDataCondition();
        devicePropertyConditionConditions.withDeviceId("07b69d78-c716-4be6-9545-869920738397")
            .withProductId("074abacf-cdb1-4f52-8c88-5864e50d332c")
            .withFilters(listDevicePropertyConditionFilters);
        List<RuleCondition> listConditionGroupConditions = new ArrayList<>();
        listConditionGroupConditions.add(
            new RuleCondition()
                .withType("DEVICE_DATA")
                .withDevicePropertyCondition(devicePropertyConditionConditions)
                .withSimpleTimerCondition(simpleTimerConditionConditions)
                .withDailyTimerCondition(dailyTimerConditionConditions)
        );
        ConditionGroup conditionGroupbody = new ConditionGroup();
        conditionGroupbody.withConditions(listConditionGroupConditions)
            .withLogic("or")
            .withTimeRange(timeRangeConditionGroup);
        body.withAppId("string");
        body.withStatus("string");
        body.withRuleType("DEVICE_LINKAGE");
        body.withActions(listbodyActions);
        body.withConditionGroup(conditionGroupbody);
        body.withDescription("string");
        body.withName("openLight");
        request.withBody(body);
        try {
            UpdateRuleResponse response = client.updateRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改联动规则，在周二的18点至24点运行规则，当满足visibility小于30或者到19点或者到达指定时触发告警设备属异常，并下发设备命令。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```



```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：    .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateRuleRequest()
        deviceAlarmActions = ActionDeviceAlarm(
            name="设备属性异常",
            alarm_status="fault",
            severity="warning",
            description="****设备属性异常。"
        )
        smnForwardingActions = ActionSmnForwarding(
            region_name="region_name",
            project_id="project_id",
            theme_name="theme_name",
            topic_urn="topic_urn",
            message_content="message_content",
            message_title="message_title"
        )
        cmdDeviceCommand = Cmd(
            command_name="SET_LIGHT_SWITCH",
            command_body="{\"value\":0}",
            service_id="Switch",
            buffer_timeout=0,
            response_timeout=1800,
            mode="string"
        )
        deviceCommandActions = ActionDeviceCommand(
            device_id="3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
            cmd=cmdDeviceCommand
        )
        listActionsbody = [
            RuleAction(
                type="DEVICE_CMD",
                device_command=deviceCommandActions,
                smn_forwarding=smnForwardingActions,
                device_alarm=deviceAlarmActions
            )
        ]
        timeRangeConditionGroup = TimeRange(
            start_time="18:00",
            end_time="23:00",
            days_of_week="2"
        )
        dailyTimerConditionConditions = DailyTimerType(
            time="19:00",
            days_of_week="2"
        )
        simpleTimerConditionConditions = SimpleTimerType(
            start_time="20190122T141500Z",
            repeat_interval=1,
            repeat_count=1
        )
        strategyFilters = Strategy(
            trigger="reverse",
            event_valid_time=300
        )
        listFiltersDevicePropertyCondition = [
            PropertyFilter(
                path="StreetLight/visibility",
                operator="<",
                value="30",
                strategy=strategyFilters
            )
        ]
```

```
        devicePropertyConditionConditions = DeviceDataCondition(
            device_id="07b69d78-c716-4be6-9545-869920738397",
            product_id="074abacf-cdb1-4f52-8c88-5864e50d332c",
            filters=listFiltersDevicePropertyCondition
        )
        listConditionsConditionGroup = [
            RuleCondition(
                type="DEVICE_DATA",
                device_property_condition=devicePropertyConditionConditions,
                simple_timer_condition=simpleTimerConditionConditions,
                daily_timer_condition=dailyTimerConditionConditions
            )
        ]
        conditionGroupbody = ConditionGroup(
            conditions=listConditionsConditionGroup,
            logic="or",
            time_range=timeRangeConditionGroup
        )
        request.body = Rule(
            app_id="string",
            status="string",
            rule_type="DEVICE_LINKAGE",
            actions=listActionsbody,
            condition_group=conditionGroupbody,
            description="string",
            name="openLight"
        )
        response = client.update_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改联动规则，在周二的18点至24点运行规则，当满足visibility小于30或者到19点或者到达指定时触发告警设备属异常，并下发设备命令。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```

```
client := iotda.NewIoTDAClient(
    iotda.IoTDAClientBuilder().
        // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
        WithRegion(region.NewRegion("cn-north-4", endpoint)).
        WithCredential(auth).
        Build())

request := &model.UpdateRuleRequest{}
descriptionDeviceAlarm:= "****设备属性异常。"
deviceAlarmActions := &model.ActionDeviceAlarm{
    Name: "设备属性异常",
    AlarmStatus: "fault",
    Severity: "warning",
    Description: &descriptionDeviceAlarm,
}
messageContentSmnForwarding:= "message_content"
smnForwardingActions := &model.ActionSmnForwarding{
    RegionName: "region_name",
    ProjectId: "project_id",
    ThemeName: "theme_name",
    TopicUrn: "topic_urn",
    MessageContent: &messageContentSmnForwarding,
    MessageTitle: "message_title",
}
var commandBodyCmd interface{} = "{\"value\":0}"
bufferTimeoutCmd:= int32(0)
responseTimeoutCmd:= int32(1800)
modeCmd:= "string"
cmdDeviceCommand := &model.Cmd{
    CommandName: "SET_LIGHT_SWITCH",
    CommandBody: &commandBodyCmd,
    ServiceId: "Switch",
    BufferTimeout: &bufferTimeoutCmd,
    ResponseTimeout: &responseTimeoutCmd,
    Mode: &modeCmd,
}
deviceIdDeviceCommand:= "3a9e52d9-3ebf-4985-89e9-6d2396748a2f"
deviceCommandActions := &model.ActionDeviceCommand{
    DeviceId: &deviceIdDeviceCommand,
    Cmd: cmdDeviceCommand,
}
var listActionsbody = []model.RuleAction{
    {
        Type: "DEVICE_CMD",
        DeviceCommand: deviceCommandActions,
        SmnForwarding: smnForwardingActions,
        DeviceAlarm: deviceAlarmActions,
    },
}
daysOfWeekTimeRange:= "2"
timeRangeConditionGroup := &model.TimeRange{
    StartTime: "18:00",
    EndTime: "23:00",
    DaysOfWeek: &daysOfWeekTimeRange,
}
daysOfWeekDailyTimerCondition:= "2"
dailyTimerConditionConditions := &model.DailyTimerType{
    Time: "19:00",
    DaysOfWeek: &daysOfWeekDailyTimerCondition,
}
simpleTimerConditionConditions := &model.SimpleTimerType{
    StartTime: "20190122T141500Z",
    RepeatInterval: int32(1),
    RepeatCount: int32(1),
}
triggerStrategy:= "reverse"
eventValidTimeStrategy:= int32(300)
strategyFilters := &model.Strategy{
```



```
        Trigger: &triggerStrategy,
        EventValidTime: &eventValidTimeStrategy,
    }
    valueFilters:= "30"
    var listFiltersDevicePropertyCondition = []model.PropertyFilter{
        {
            Path: "StreetLight/visibility",
            Operator: "<",
            Value: &valueFilters,
            Strategy: strategyFilters,
        },
    }
    deviceIdDevicePropertyCondition:= "07b69d78-c716-4be6-9545-869920738397"
    productIdDevicePropertyCondition:= "074abacf-cdb1-4f52-8c88-5864e50d332c"
    devicePropertyConditionConditions := &model.DeviceDataCondition{
        DeviceId: &deviceIdDevicePropertyCondition,
        ProductId: &productIdDevicePropertyCondition,
        Filters: &listFiltersDevicePropertyCondition,
    }
    var listConditionsConditionGroup = []model.RuleCondition{
        {
            Type: "DEVICE_DATA",
            DevicePropertyCondition: devicePropertyConditionConditions,
            SimpleTimerCondition: simpleTimerConditionConditions,
            DailyTimerCondition: dailyTimerConditionConditions,
        },
    }
    logicConditionGroup:= "or"
    conditionGroupbody := &model.ConditionGroup{
        Conditions: &listConditionsConditionGroup,
        Logic: &logicConditionGroup,
        TimeRange: timeRangeConditionGroup,
    }
    appIdRule:= "string"
    statusRule:= "string"
    descriptionRule:= "string"
    request.Body = &model.Rule{
        AppId: &appIdRule,
        Status: &statusRule,
        RuleType: "DEVICE_LINKAGE",
        Actions: listActionsbody,
        ConditionGroup: conditionGroupbody,
        Description: &descriptionRule,
        Name: "openLight",
    }
    response, err := client.UpdateRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request

状态码	描述
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.10.4 查询规则

功能介绍

应用服务器可调用此接口查询物联网平台中指定规则的配置信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/rules/{rule_id}

表 1-447 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID**。
rule_id	是	String	**参数说明**：规则ID，用于唯一标识一条规则，在创建规则时由物联网平台分配获得。 **取值范围**：长度不超过32，只允许字母、数字的组合。

请求参数

表 1-448 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-449 响应 Body 参数

参数	参数类型	描述
rule_id	String	规则id。 最大长度：**128**
name	String	规则名称。 最小长度：**1** 最大长度：**128**
description	String	规则的描述信息。 最大长度：**256**
condition_group	**ConditionGroup** object	规则的条件组，包含简单规则和复杂规则集合。
actions	Array of **RuleAction** objects	规则的动作列表，单个规则最多支持设置10个动作。
rule_type	String	规则的类型 • DEVICE_LINKAGE：云端联动规则。 • DEVICE_SIDE：端侧规则。

参数	参数类型	描述
status	String	规则的状态，默认值：active。 ● active：激活。 ● inactive：未激活。
app_id	String	资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的规则归属到哪个资源空间下，否则创建的规则将会归属到**默认资源空间**下。
edge_node_ids	Array of strings	归属边缘侧节点设备ID列表。
last_update_time	String	规则最后更新时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。
device_side	**DeviceSide** object	**参数说明**：端侧执行下发设备信息，当规则类型为DEVICE_SIDE时，该参数必填。

表 1-450 ConditionGroup

参数	参数类型	描述
conditions	Array of **RuleCondition** objects	**参数说明**：规则的条件列表，单个规则最多支持设置10个条件。
logic	String	**参数说明**：规则条件列表中多个条件之间的逻辑关系，默认值：and。 **取值范围**： ● and：逻辑且。 ● or：逻辑或。
time_range	**TimeRange** object	**参数说明**：规则条件触发的有效时间段。

表 1-451 RuleCondition

参数	参数类型	描述
type	String	**参数说明**：规则条件的类型。 **取值范围**： ● DEVICE_DATA：设备属性数据类型条件。 ● SIMPLE_TIMER：简单定时类型条件。 ● DAILY_TIMER：每日定时类型条件。 ● DEVICE_LINKAGE_STATUS：设备状态类型条件。

参数	参数类型	描述
device_property_condition	**DeviceDataCondition** object	**参数说明**：条件中设备数据类型的信息，当type为DEVICE_DATA时，为必选参数。
simple_timer_condition	**SimpleTimerType** object	**参数说明**：条件中简单定时类型的信息，当type为SIMPLE_TIMER时，为必选参数。
daily_timer_condition	**DailyTimerType** object	**参数说明**：条件中每日定时类型的信息，当type为DAILY_TIMER时，为必选参数。
device_linkage_status_condition	**DeviceLinkageStatusCondition** object	**参数说明**：条件中设备状态类型的信息，当规则类型为DEVICE_LINKAGE且type为DEVICE_LINKAGE_STATUS时，为必选参数。

表 1-452 DeviceDataCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备属性触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备属性触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
filters	Array of **PropertyFilter** objects	数据过滤条件。

表 1-453 PropertyFilter

参数	参数类型	描述
path	String	**参数说明**：设备属性的路径信息，格式：service_id/DataProperty，例如门磁状态为“DoorWindow/status”。 最大长度：**128**

参数	参数类型	描述
operator	String	**参数说明**：数据比较的操作符。 **取值范围**：当前支持的操作符有：>，<，>=，<=，=，in:表示在指定值中匹配和between:表示数值区间。
value	String	**参数说明**：数据比较表达式的右值。与数据比较操作符between联用时，右值表示最小值和最大值，用逗号隔开，如“20,30”表示大于等于20小于30。 最大长度：**64**
in_values	Array of strings	**参数说明**：当operator为in时该字段必填，使用该字段传递比较表达式右值，上限为20个。 最大长度：**128**
strategy	**Strategy** object	**参数说明**：规则条件的处理策略，用于确定规则是否判断上次数据是否满足条件。当rule_type为DEVICE_LINKAGE时，该参数值不能为空。端侧执行不支持该字段。

表 1-454 Strategy

参数	参数类型	描述
trigger	String	**参数说明**：规则条件触发的判断策略，默认为pulse。 **取值范围**： • pulse：设备上报的数据满足条件则触发，不判断上一次上报的数据。 • reverse：设备上一次上报的数据不满足条件，本次上报的数据满足条件则触发。
event_valid_time	Integer	**参数说明**：设备数据的有效时间，单位为秒，设备数据的产生时间以上报数据中的eventTime为基准。 最小值：**-1**

表 1-455 SimpleTimerType

参数	参数类型	描述
start_time	String	**参数说明**：规则触发的开始时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'。 最大长度：**128**

参数	参数类型	描述
repeat_interval	Integer	**参数说明**：规则触发的重复时间间隔，单位为秒。 最小值：**1** 最大值：**31536000**
repeat_count	Integer	**参数说明**：规则触发的重复次数。 最小值：**1** 最大值：**9999**

表 1-456 DailyTimerType

参数	参数类型	描述
time	String	**参数说明**：规则触发的时间，格式：HH:MM。 最大长度：**128**
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。 最大长度：**128**

表 1-457 DeviceLinkageStatusCondition

参数	参数类型	描述
device_id	String	**参数说明**：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。存在该参数时设备状态触发根据指定设备触发，该参数值和product_id不能同时为空。如果该参数和product_id同时存在时，以该参数值对应的设备进行条件过滤。 **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。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。存在该参数且device_id为空时设备状态触发匹配该产品下所有设备触发，该参数值和device_id不能同时为空。 最大长度：**128**
status_list	Array of strings	**参数说明**：状态列表，设备状态条件携带该参数。 **取值范围**： • ONLINE：设备上线 • OFFLINE：设备下线 最大长度：**128**

参数	参数类型	描述
duration	Integer	**持续时长**：设备状态持续时长，取值范围:0-60(分钟)。 最小值：**0** 最大值：**60** 缺省值：**0**

表 1-458 TimeRange

参数	参数类型	描述
start_time	String	**参数说明**：规则条件触发的开始时间，格式：HH:mm。
end_time	String	**参数说明**：规则条件触发的结束时间，格式：HH:mm。若结束时间与开始时间一致，则时间为全天。
days_of_week	String	**参数说明**：星期列表，以逗号分隔。1代表周日，2代表周一，依次类推，默认为每天。星期列表中的日期为开始时间的日期。

表 1-459 RuleAction

参数	参数类型	描述
type	String	**参数说明**：规则动作的类型，端侧执行只支持下发设备命令消息类型。 **取值范围**： • DEVICE_CMD：下发设备命令消息类型。 • SMN_FORWARDING：发送SMN消息类型。 • DEVICE_ALARM：上报设备告警消息类型。当选择该类型时，condition中必须有DEVICE_DATA条件类型。该类型动作只能唯一。
device_command	**ActionDeviceCommand** object	下发设备命令消息内容。当type为DEVICE_CMD时，必填。
smn_forwarding	**ActionSmnForwarding** object	发送给SMN消息结构。当规则类型为DEVICE_LINKAGE且type为SMN_FORWARDING时，必填。
device_alarm	**ActionDeviceAlarm** object	上报设备告警消息内容。当规则类型为DEVICE_LINKAGE且type为DEVICE_ALARM时，必填。

表 1-460 ActionDeviceCommand

参数	参数类型	描述
device_id	String	**参数说明**：下发命令的设备ID。 ● 当创建设备数据规则时，若device_id为空，则命令下发给触发条件的设备。 ● 当创建定时规则时，不允许为空。 **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。
cmd	**CMD** object	**参数说明**：下发的命令信息。

表 1-461 CMD

参数	参数类型	描述
command_name	String	**参数说明**：设备命令名称，在设备关联的产品模型中定义。 最大长度：**128**
command_body	Object	**参数说明**：设备命令参数，Json格式。 使用LWM2M协议设备命令示例：{"value":"1"}，里面是一个个键值对，每个键都是产品模型中命令的参数名（ paraName ）。 使用MQTT协议设备命令示例：{"header": {"mode": "ACK","from": "/users/testUser","method": "SET_TEMPERATURE_READ_PERIOD","to":"/devices/{device_id}/services/{service_id}"},"body": {"value" : "1"}}。 ● mode：必选，设备收到命令后是否需要回复确认消息，默认为ACK模式。ACK表示需要回复确认消息，NOACK表示不需要回复确认消息，其它值无效。 ● from：可选，命令发送方的地址。App发起请求时格式为/users/{userId} ，应用服务器发起请求时格式为/{serviceName}，物联网平台发起请求时格式为/cloud/{serviceName}。 ● to：可选，命令接收方的地址，格式为/devices/{device_id}/services/{service_id}。 ● method：可选，产品模型中定义的命令名称。 ● body：可选，命令的消息体，里面是一个个键值对，每个键都是产品模型中命令的参数名（ paraName ）。具体格式需要应用和设备约定。

参数	参数类型	描述
service_id	String	**参数说明**：设备命令所属的设备服务ID，在设备关联的产品模型中定义。 最大长度：**64**
buffer_timeout	Integer	**参数说明**：设备命令的缓存时间，单位为秒，表示物联网平台在把命令下发给设备前缓存命令的有效时间，超过这个时间后命令将不再下发，默认值为172800s（48小时）。 如果buffer_timeout设置为0，则无论物联网平台上设置的命令下发模式是什么，该命令都会立即下发给设备。 最小值：**0** 最大值：**31536000** 缺省值：**172800**
response_timeout	Integer	**参数说明**：命令响应的有效时间，单位为秒，表示设备接收到命令后，在response_timeout时间内响应有效，超过这个时间未收到命令的响应，则认为命令响应超时，默认值为1800s。 最小值：**1** 最大值：**31536000** 缺省值：**1800**
mode	String	**参数说明**：设备命令的下发模式，仅当buffer_timeout的值大于0时有效。 ● ACTIVE：主动模式，物联网平台主动将命令下发给设备。 ● PASSIVE：被动模式，物联网平台创建设备命令后，会直接缓存命令。等到设备再次上线或者上报上一条命令的执行结果后才下发命令。

表 1-462 ActionSmnForwarding

参数	参数类型	描述
region_name	String	**参数说明**：SMN服务对应的region区域。 最大长度：**256**
project_id	String	**参数说明**：SMN服务对应的projectId信息。
theme_name	String	**参数说明**：SMN服务对应的主题名称。 最大长度：**256**
topic_urn	String	**参数说明**：SMN服务对应的topic的主题URN。 最大长度：**256**

参数	参数类型	描述
message_content	String	**参数说明**：短信或邮件的内容。 最大长度：**1024**
message_template_name	String	**参数说明**：SMN服务对应的模板名称。
message_title	String	**参数说明**：短信或邮件的主题。最大长度支持UTF-8编码后的521个字节。

表 1-463 ActionDeviceAlarm

参数	参数类型	描述
name	String	**参数说明**：告警名称。 最大长度：**256**
alarm_status	String	**参数说明**：告警状态。 **取值范围**： ● fault：上报告警。 ● recovery：恢复告警。
severity	String	**参数说明**：告警级别。 **取值范围**：warning（警告）、minor（一般）、major（严重）和critical（致命）。
dimension	String	**参数说明**：告警维度，与告警名称和告警级别组合起来共同标识一条告警，默认不携带该字段为用户维度告警，支持设备维度和资源空间维度告警。 **取值范围**： ● device：设备维度。 ● app：资源空间维度。
description	String	**参数说明**：告警的描述信息。 最大长度：**256**

表 1-464 DeviceSide

参数	参数类型	描述
device_ids	Array of strings	**参数说明**：端侧执行下发的目标设备ID列表。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。 最大长度：**128**

请求示例

查询设备规则。

```
GET https://{endpoint}/v5/iot/{project_id}/rules/{rule_id}
```

响应示例

状态码： 200

OK

```
{
  "rule_id" : "5eb3628d017d9105d0cf9aec",
  "name" : "openLight",
  "condition_group" : {
    "conditions" : [ {
      "type" : "DEVICE_DATA",
      "device_property_condition" : {
        "device_id" : "07b69d78-c716-4be6-9545-869920738397",
        "filters" : [ {
          "path" : "StreetLight/visibility",
          "operator" : "<",
          "value" : "30",
          "strategy" : {
            "trigger" : "reverse",
            "event_valid_time" : 300
          }
        } ]
      }
    } ],
    "logic" : "and"
  },
  "actions" : [ {
    "type" : "DEVICE_CMD",
    "device_command" : {
      "device_id" : "3a9e52d9-3ebf-4985-89e9-6d2396748a2f",
      "cmd" : {
        "command_name" : "SET_LIGHT_SWITCH",
        "command_body" : {
          "value" : 0
        },
        "service_id" : "Switch",
        "buffer_timeout" : 0,
        "response_timeout" : 1800
      }
    }
  } ],
  "rule_type" : "DEVICE_LINKAGE",
  "status" : "active",
  "app_id" : "9562bf8541e44361b6ae3a7e9fbe1144",
  "last_update_time" : "20221017T025425Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
```

```
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowRuleRequest request = new ShowRuleRequest();
        try {
            ShowRuleResponse response = client.showRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
```

```
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowRuleRequest()
        response = client.show_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowRuleRequest{}
    response, err := client.ShowRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```



```
  }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.10.5 删除规则

功能介绍

应用服务器可调用此接口删除物联网平台中的指定规则。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/rules/{rule_id}

表 1-465 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID**。

参数	是否必选	参数类型	描述
rule_id	是	String	**参数说明**：规则ID，用于唯一标识一条规则，在创建规则时由物联网平台分配获得。 **取值范围**：长度不超过32，只允许字母、数字的组合。

请求参数

表 1-466 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除设备规则。

```
DELETE https://{endpoint}/v5/iot/{project_id}/rules/{rule_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteRuleSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteRuleRequest request = new DeleteRuleRequest();
        try {
            DeleteRuleResponse response = client.deleteRule(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *
```

```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteRuleRequest()
        response = client.delete_rule(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
```

```
        Build())

    request := &model.DeleteRuleRequest{}
    response, err := client.DeleteRule(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.10.6 修改规则状态

功能介绍

应用服务器可调用此接口修改物联网平台中指定规则的状态，激活或者去激活规则。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/rules/{rule_id}/status

表 1-467 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID**。
rule_id	是	String	**参数说明**：规则Id。 **取值范围**：长度不超过32，只允许字母、数字的组合。

请求参数

表 1-468 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-469 请求 Body 参数

参数	是否必选	参数类型	描述
status	是	String	**参数说明**：规则的激活状态。 **取值范围**： • active：激活。 • inactive：未激活。

响应参数

状态码： 200

表 1-470 响应 Body 参数

参数	参数类型	描述
status	String	**参数说明**：规则的激活状态。 **取值范围**： ● active：激活。 ● inactive：未激活。

请求示例

修改规则状态，修改状态为激活。

```
PUT https://{endpoint}/v5/iot/{project_id}/rules/{rule_id}/status

{
  "status" : "active"
}
```

响应示例

状态码： 200

OK

```
{
  "status" : "active"
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改规则状态，修改状态为激活。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ChangeRuleStatusSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
```



```
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ChangeRuleStatusRequest request = new ChangeRuleStatusRequest();
        RuleStatus body = new RuleStatus();
        body.withStatus("active");
        request.withBody(body);
        try {
            ChangeRuleStatusResponse response = client.changeRuleStatus(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改规则状态，修改状态为激活。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()
```



```
    try:
        request = ChangeRuleStatusRequest()
        request.body = RuleStatus(
            status="active"
        )
        response = client.change_rule_status(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改规则状态，修改状态为激活。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ChangeRuleStatusRequest{}
    request.Body = &model.RuleStatus{
        Status: "active",
    }
    response, err := client.ChangeRuleStatus(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.11 设备影子

设备影子介绍：

设备影子是一个用于存储和检索设备当前状态信息的JSON文档。

- 每个设备有且只有一个设备影子，由设备ID唯一标识

- 设备影子用于存储设备上报的属性和应用程序期望的设备属性

- 无论该设备是否在线，都可以通过该影子获取和设置设备的状态

1.4.11.1 查询设备影子数据

功能介绍

应用服务器可调用此接口查询指定设备的设备影子信息，包括对设备的期望属性信息（desired区）和设备最新上报的属性信息（reported区）。

设备影子介绍： 设备影子是一个用于存储和检索设备当前状态信息的JSON文档。

- 每个设备有且只有一个设备影子，由设备ID唯一标识
- 设备影子用于存储设备上报的(状态)属性和应用程序期望的设备(状态)属性
- 无论该设备是否在线，都可以通过该影子获取和设置设备的属性
- 设备上线或者设备上报属性时，如果desired区和reported区存在差异，则将差异部分下发给设备，配置的预期属性需在产品模型中定义且method具有可写属性“W”才可下发

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。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/devices/{device_id}/shadow

表 1-471 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。 **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。

请求参数

表 1-472 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-473 响应 Body 参数

参数	参数类型	描述
device_id	String	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。
shadow	Array of **DeviceShadowData** objects	设备影子数据结构体。

表 1-474 DeviceShadowData

参数	参数类型	描述
service_id	String	设备的服务ID，在设备关联的产品模型中定义。 最小长度：**1** 最大长度：**256**
desired	**DeviceShadowProperties** object	用户最近一次对设备下发的预期数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)。
reported	**DeviceShadowProperties** object	设备最近一次上报的属性数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)。
version	Long	设备影子的版本，携带该参数时平台会校验值必须等于当前影子版本，初始从0开始。

表 1-475 DeviceShadowProperties

参数	参数类型	描述
properties	Object	设备影子的属性数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)，目前如样例所示只支持一层结构。 **注意**：JSON结构的key当前不支持特殊字符：点(.)、dollar符号($)、空char(十六进制的ASCII码为00),key为以上特殊字符无法正常刷新设备影子

参数	参数类型	描述
event_time	String	事件操作时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。 最大长度：**256**

请求示例

查询指定设备的设备影子。

```
GET https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/shadow
```

响应示例

状态码： 200

OK

```
{
  "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
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDeviceShadowSolution {

    public static void main(String[] args) {
```



```
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running
this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDeviceShadowRequest request = new ShowDeviceShadowRequest();
        try {
            ShowDeviceShadowResponse response = client.showDeviceShadow(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
```

```
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDeviceShadowRequest()
        response = client.show_device_shadow(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDeviceShadowRequest{}
    response, err := client.ShowDeviceShadow(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	INTERNAL SERVER ERROR

错误码

请参见**错误码**。

1.4.11.2 配置设备影子预期数据

功能介绍

应用服务器可调用此接口配置设备影子的预期属性（desired区），当设备上线或者设备上报属性时把属性下发给设备。

设备影子介绍： 设备影子是一个用于存储和检索设备当前状态信息的JSON文档。

- 每个设备有且只有一个设备影子，由设备ID唯一标识
- 设备影子用于存储设备上报的(状态)属性和应用程序期望的设备(状态)属性
- 无论该设备是否在线，都可以通过该影子获取和设置设备的属性
- 设备上线或者设备上报属性时，如果desired区和reported区存在差异，则将差异部分下发给设备，配置的预期属性需在产品模型中定义且method具有可写属性“W”才可下发
- 该接口仅支持配置单个设备的设备影子的预期数据，如需多个设备的设备影子配置，请参见 **创建批量任务**。

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。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/devices/{device_id}/shadow

表 1-476 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
device_id	是	String	**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。*

请求参数

表 1-477 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-478 请求 Body 参数

参数	是否必选	参数类型	描述
shadow	否	Array of **UpdateDesired** objects	**参数说明**：设备影子期望值构体。

表 1-479 UpdateDesired

参数	是否必选	参数类型	描述
service_id	是	String	**参数说明**：设备的服务ID，在设备关联的产品模型中定义。 最小长度：**1** 最大长度：**256**
desired	是	Object	**参数说明**：设备影子期望属性数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)，目前如样例所示只支持一层结构；如果想要删除整个desired可以填写空Object(例如"desired":{})，如果想要删除某一个属性期望值可以将属性置位null(例如{"temperature":null})
version	否	Long	**参数说明**：设备影子的版本，携带该参数时平台会校验值必须等于当前影子版本，初始从0开始。

响应参数

状态码： 200

表 1-480 响应 Body 参数

参数	参数类型	描述
device_id	String	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。
shadow	Array of **DeviceShadowData** objects	设备影子数据结构体。

表 1-481 DeviceShadowData

参数	参数类型	描述
service_id	String	设备的服务ID，在设备关联的产品模型中定义。 最小长度：**1** 最大长度：**256**
desired	**DeviceShadowProperties** object	用户最近一次对设备下发的预期数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)。
reported	**DeviceShadowProperties** object	设备最近一次上报的属性数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)。
version	Long	设备影子的版本，携带该参数时平台会校验值必须等于当前影子版本，初始从0开始。

表 1-482 DeviceShadowProperties

参数	参数类型	描述
properties	Object	设备影子的属性数据，Json格式，里面是一个个键值对，每个键都是产品模型中属性的参数名(property_name)，目前如样例所示只支持一层结构。 **注意**：JSON结构的key当前不支持特殊字符：点(.)、dollar符号($)、空char(十六进制的ASCII码为00),key为以上特殊字符无法正常刷新设备影子
event_time	String	事件操作时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。 最大长度：**256**

请求示例

- 配置设备影子预期数据，temperature的期望值为60。

```
PUT https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/shadow

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

- 删除设备影子预期数据，将设备影子中属性temperature的期望值删除。

```
PUT https://{endpoint}/v5/iot/{project_id}/devices/{device_id}/shadow
```



```
{
  "shadow" : [ {
    "service_id" : "WaterMeter",
    "desired" : {
      "temperature" : null
    },
    "version" : 2
  } ]
}
```

响应示例

状态码： 200

OK

```
{
  "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" : 2
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 配置设备影子预期数据，temperature的期望值为60。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateDeviceShadowDesiredDataSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
```

```
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateDeviceShadowDesiredDataRequest request = new
UpdateDeviceShadowDesiredDataRequest();
        UpdateDesireds body = new UpdateDesireds();
        List<UpdateDesired> listbodyShadow = new ArrayList<>();
        listbodyShadow.add(
            new UpdateDesired()
                .withServiceId("WaterMeter")
                .withDesired("{\"temperature\":\"60\"}")
                .withVersion(1L)
        );
        body.withShadow(listbodyShadow);
        request.withBody(body);
        try {
            UpdateDeviceShadowDesiredDataResponse response =
client.updateDeviceShadowDesiredData(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 删除设备影子预期数据，将设备影子中属性temperature的期望值删除。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateDeviceShadowDesiredDataSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
```

```
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateDeviceShadowDesiredDataRequest request = new
UpdateDeviceShadowDesiredDataRequest();
        UpdateDesireds body = new UpdateDesireds();
        List<UpdateDesired> listbodyShadow = new ArrayList<>();
        listbodyShadow.add(
            new UpdateDesired()
                .withServiceId("WaterMeter")
                .withDesired(new Object())
                .withVersion(2L)
        );
        body.withShadow(listbodyShadow);
        request.withBody(body);
        try {
            UpdateDeviceShadowDesiredDataResponse response =
client.updateDeviceShadowDesiredData(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 配置设备影子预期数据，temperature的期望值为60。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
```

```
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateDeviceShadowDesiredDataRequest()
        listShadowbody = [
            UpdateDesired(
                service_id="WaterMeter",
                desired="{\"temperature\":\"60\"}",
                version=1
            )
        ]
        request.body = UpdateDesireds(
            shadow=listShadowbody
        )
        response = client.update_device_shadow_desired_data(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 删除设备影子预期数据，将设备影子中属性temperature的期望值删除。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
```

```
        request = UpdateDeviceShadowDesiredDataRequest()
        listShadowbody = [
            UpdateDesired(
                service_id="WaterMeter",
                desired={},
                version=2
            )
        ]
        request.body = UpdateDesireds(
            shadow=listShadowbody
        )
        response = client.update_device_shadow_desired_data(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 配置设备影子预期数据，temperature的期望值为60。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateDeviceShadowDesiredDataRequest{}
    desiredShadow:= "{\"temperature\":\"60\"}"
    var desiredShadowInterface interface{} = desiredShadow
    versionShadow:= int64(1)
    var listShadowbody = []model.UpdateDesired{
        {
            ServiceId: "WaterMeter",
            Desired: &desiredShadowInterface,
            Version: &versionShadow,
```



```
        },
    }
    request.Body = &model.UpdateDesireds{
        Shadow: &listShadowbody,
    }
    response, err := client.UpdateDeviceShadowDesiredData(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 删除设备影子预期数据，将设备影子中属性temperature的期望值删除。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateDeviceShadowDesiredDataRequest{}
    desiredShadow:= make(map[string]string)
    var desiredShadowInterface interface{} = desiredShadow
    versionShadow:= int64(2)
    var listShadowbody = []model.UpdateDesired{
        {
            ServiceId: "WaterMeter",
            Desired: &desiredShadowInterface,
            Version: &versionShadow,
        },
    }
    request.Body = &model.UpdateDesireds{
        Shadow: &listShadowbody,
    }
    response, err := client.UpdateDeviceShadowDesiredData(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
```

```
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
409	CONFLICT
500	INTERNAL SERVER ERROR

错误码

请参见**错误码**。

1.4.12 设备组管理

1.4.12.1 添加设备组

功能介绍

应用服务器可调用此接口新建设备组，一个华为云账号下最多可有1,000个设备组，包括父设备组和子设备组。设备组的最大层级关系不超过5层，即群组形成的关系树最大深度不超过5。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-group

表 1-483 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-484 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-485 请求 Body 参数

参数	是否必选	参数类型	描述
name	否	String	**参数说明**：设备组名称，单个资源空间下不可重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**64**
description	否	String	**参数说明**：设备组描述。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**64**

参数	是否必选	参数类型	描述
super_group_id	否	String	**参数说明**：父设备组ID，携带该参数时表示在该设备组下创建一个子设备组,动态群组不支持该参数。 **取值范围**：长度不超过36，十六进制字符串和连接符（-）的组合。
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的设备组归属到哪个资源空间下，否则创建的设备组将会归属到**默认资源空间**下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
group_type	否	String	**参数说明**：设备组类型，默认为静态设备组；当设备组类型为动态设备组时，需要填写动态设备组组规则
dynamic_group_rule	否	String	**参数说明**：动态设备组规则语法和高级搜索保持一致，只需要填写where 子句内容，其余子句无需填写，todo补充说明

响应参数

状态码： 201

表 1-486 响应 Body 参数

参数	参数类型	描述
group_id	String	设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。
name	String	设备组名称，单个资源空间下不可重复。
description	String	设备组描述。
super_group_id	String	父设备组ID，该设备组的父设备组ID。
group_type	String	设备组类型，分为动态设备组和静态设备组两种
dynamic_group_rule	String	动态设备组规则

请求示例

- 创建静态设备组，设备组名为GroupA。

```
POST https://{endpoint}/v5/iot/{project_id}/device-group

{
  "name" : "GroupA",
  "description" : "群组A",
  "super_group_id" : "04ed32dc1b0025b52fe3c01a27c2b0a8",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "group_type" : "STATIC"
}
```

- 创建动态设备组，设备组名为GroupA。

```
POST https://{endpoint}/v5/iot/{project_id}/device-group

{
  "name" : "GroupA",
  "description" : "群组A",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "dynamic_group_rule" : "product_id = '63fef97897bacf7a56438cba'",
  "group_type" : "DYNAMIC"
}
```

响应示例

状态码： 201

Created

```
{
  "group_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
  "name" : "GroupA",
  "description" : "群组A",
  "super_group_id" : "04ed32dc1b0025b52fe3c01a27c2b0a8",
  "group_type" : "STATIC",
  "dynamic_group_rule" : null
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建静态设备组，设备组名为GroupA。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class AddDeviceGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
```



```
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddDeviceGroupRequest request = new AddDeviceGroupRequest();
        AddDeviceGroupDTO body = new AddDeviceGroupDTO();
        body.withGroupType("STATIC");
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withSuperGroupId("04ed32dc1b0025b52fe3c01a27c2b0a8");
        body.withDescription("群组A");
        body.withName("GroupA");
        request.withBody(body);
        try {
            AddDeviceGroupResponse response = client.addDeviceGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建动态设备组，设备组名为GroupA。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class AddDeviceGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
```

```
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddDeviceGroupRequest request = new AddDeviceGroupRequest();
        AddDeviceGroupDTO body = new AddDeviceGroupDTO();
        body.withDynamicGroupRule("product_id = '63fef97897bacf7a56438cba'");
        body.withGroupType("DYNAMIC");
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withDescription("群组A");
        body.withName("GroupA");
        request.withBody(body);
        try {
            AddDeviceGroupResponse response = client.addDeviceGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建静态设备组，设备组名为GroupA。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddDeviceGroupRequest()
        request.body = AddDeviceGroupDTO(
            group_type="STATIC",
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            super_group_id="04ed32dc1b0025b52fe3c01a27c2b0a8",
            description="群组A",
            name="GroupA"
        )
        response = client.add_device_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建动态设备组，设备组名为GroupA。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddDeviceGroupRequest()
        request.body = AddDeviceGroupDTO(
            dynamic_group_rule="product_id = '63fef97897bacf7a56438cba'",
            group_type="DYNAMIC",
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            description="群组A",
            name="GroupA"
        )
        response = client.add_device_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
```

```
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建静态设备组，设备组名为GroupA。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddDeviceGroupRequest{}
    groupTypeAddDeviceGroupDto:= "STATIC"
    appIdAddDeviceGroupDto:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    superGroupIdAddDeviceGroupDto:= "04ed32dc1b0025b52fe3c01a27c2b0a8"
    descriptionAddDeviceGroupDto:= "群组A"
    nameAddDeviceGroupDto:= "GroupA"
    request.Body = &model.AddDeviceGroupDto{
        GroupType: &groupTypeAddDeviceGroupDto,
        AppId: &appIdAddDeviceGroupDto,
        SuperGroupId: &superGroupIdAddDeviceGroupDto,
        Description: &descriptionAddDeviceGroupDto,
        Name: &nameAddDeviceGroupDto,
    }
    response, err := client.AddDeviceGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建动态设备组，设备组名为GroupA。

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddDeviceGroupRequest{}
    dynamicGroupRuleAddDeviceGroupDto:= "product_id = '63fef97897bacf7a56438cba'"
    groupTypeAddDeviceGroupDto:= "DYNAMIC"
    appIdAddDeviceGroupDto:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    descriptionAddDeviceGroupDto:= "群组A"
    nameAddDeviceGroupDto:= "GroupA"
    request.Body = &model.AddDeviceGroupDto{
        DynamicGroupRule: &dynamicGroupRuleAddDeviceGroupDto,
        GroupType: &groupTypeAddDeviceGroupDto,
        AppId: &appIdAddDeviceGroupDto,
        Description: &descriptionAddDeviceGroupDto,
        Name: &nameAddDeviceGroupDto,
    }
    response, err := client.AddDeviceGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.12.2 查询设备组列表

功能介绍

应用服务器可调用此接口查询物联网平台中的设备组信息列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-group

表 1-487 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-488 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**
last_modified_time	否	String	**参数说明**：查询设备组在last_modified_time之后修改的记录。 **取值范围**：格式为yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的产品列表，不携带该参数则会查询该用户下所有产品列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
group_type	否	String	**参数说明**：设备组类型，默认为静态设备组；当设备组类型为动态设备组时，需要填写动态设备组规则
name	否	String	**参数说明**：设备组名称，单个资源空间下不可重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**64**

请求参数

表 1-489 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-490 响应 Body 参数

参数	参数类型	描述
device_groups	Array of **DeviceGroupResponseSummary** objects	设备组信息列表。
page	**Page** object	查询结果的分页信息。

表 1-491 DeviceGroupResponseSummary

参数	参数类型	描述
group_id	String	设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。
name	String	设备组名称，单个资源空间下不可重复。
description	String	设备组描述。
super_group_id	String	父设备组ID，该设备组的父设备组ID。
group_type	String	**参数说明**：设备组类型，默认为静态设备组；当设备组类型为动态设备组时，需要填写动态设备组规则

表 1-492 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

列表查询设备组。

```
GET https://{endpoint}/v5/iot/{project_id}/device-group
```

响应示例

状态码： 200

OK

```
{
  "device_groups" : [ {
    "group_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
    "name" : "GroupA",
    "description" : "群组A",
    "super_group_id" : "04ed32dc1b0025b52fe3c01a27c2b0a8",
    "group_type" : "STATIC"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListDeviceGroupsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListDeviceGroupsRequest request = new ListDeviceGroupsRequest();
        request.withLimit(<limit>);
```



```
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        request.withLastModifiedTime("<last_modified_time>");
        request.withAppId("<app_id>");
        request.withGroupType("<group_type>");
        request.withName("<name>");
        try {
            ListDeviceGroupsResponse response = client.listDeviceGroups(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListDeviceGroupsRequest()
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        request.last_modified_time = "<last_modified_time>"
        request.app_id = "<app_id>"
        request.group_type = "<group_type>"
        request.name = "<name>"
        response = client.list_device_groups(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
```



```
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListDeviceGroupsRequest{}
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    lastModifiedTimeRequest:= "<last_modified_time>"
    request.LastModifiedTime = &lastModifiedTimeRequest
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    groupTypeRequest:= "<group_type>"
    request.GroupType = &groupTypeRequest
    nameRequest:= "<name>"
    request.Name = &nameRequest
    response, err := client.ListDeviceGroups(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.12.3 查询设备组

功能介绍

应用服务器可调用此接口查询指定设备组详情。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-group/{group_id}

表 1-493 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
group_id	是	String	**参数说明**：设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。 **取值范围**：长度不超过36，十六进制字符串和连接符（-）的组合。

请求参数

表 1-494 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-495 响应 Body 参数

参数	参数类型	描述
group_id	String	设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。
name	String	设备组名称，单个资源空间下不可重复。
description	String	设备组描述。
super_group_id	String	父设备组ID，该设备组的父设备组ID。
group_type	String	设备组类型，分为动态设备组和静态设备组两种
dynamic_group_rule	String	动态设备组规则

请求示例

查询指定设备组详情。

```
GET https://{endpoint}/v5/iot/{project_id}/device-group/{group_id}
```

响应示例

状态码： 200

OK

```
{
  "group_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
  "name" : "GroupA",
  "description" : "群组A",
  "super_group_id" : "04ed32dc1b0025b52fe3c01a27c2b0a8",
  "group_type" : "STATIC"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDeviceGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDeviceGroupRequest request = new ShowDeviceGroupRequest();
        try {
            ShowDeviceGroupResponse response = client.showDeviceGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
```



```
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDeviceGroupRequest()
        response = client.show_device_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
```

```
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDeviceGroupRequest{}
    response, err := client.ShowDeviceGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
404	Not Found
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.12.4 修改设备组

功能介绍

应用服务器可调用此接口修改物联网平台中指定设备组。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/device-group/{group_id}

表 1-496 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
group_id	是	String	**参数说明**：设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。 **取值范围**：长度不超过36，十六进制字符串和连接符（ - ）的组合。

请求参数

表 1-497 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-498 请求 Body 参数

参数	是否必选	参数类型	描述
name	否	String	**参数说明**：设备组名称，单个资源空间下不可重复。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**64**
description	否	String	**参数说明**：设备组描述。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**64**

响应参数

状态码： 200

表 1-499 响应 Body 参数

参数	参数类型	描述
group_id	String	设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。
name	String	设备组名称，单个资源空间下不可重复。
description	String	设备组描述。
super_group_id	String	父设备组ID，该设备组的父设备组ID。
group_type	String	设备组类型，分为动态设备组和静态设备组两种
dynamic_group_rule	String	动态设备组规则

请求示例

修改设备组，将设备组的名称修改为GroupA。

```
PUT https://{endpoint}/v5/iot/{project_id}/device-group/{group_id}

{
  "name" : "GroupA",
  "description" : "群组A"
}
```


响应示例

状态码： 200

OK

```
{
  "group_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
  "name" : "GroupA",
  "description" : "群组A",
  "super_group_id" : "04ed32dc1b0025b52fe3c01a27c2b0a8",
  "group_type" : "STATIC",
  "dynamic_group_rule" : null
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改设备组，将设备组的名称修改为GroupA。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateDeviceGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateDeviceGroupRequest request = new UpdateDeviceGroupRequest();
        UpdateDeviceGroupDTO body = new UpdateDeviceGroupDTO();
        body.withDescription("群组A");
        body.withName("GroupA");
        request.withBody(body);
```

```
        try {
            UpdateDeviceGroupResponse response = client.updateDeviceGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改设备组，将设备组的名称修改为GroupA。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateDeviceGroupRequest()
        request.body = UpdateDeviceGroupDTO(
            description="群组A",
            name="GroupA"
        )
        response = client.update_device_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改设备组，将设备组的名称修改为GroupA。


```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateDeviceGroupRequest{}
    descriptionUpdateDeviceGroupDto:= "群组A"
    nameUpdateDeviceGroupDto:= "GroupA"
    request.Body = &model.UpdateDeviceGroupDto{
        Description: &descriptionUpdateDeviceGroupDto,
        Name: &nameUpdateDeviceGroupDto,
    }
    response, err := client.UpdateDeviceGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request

状态码	描述
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.12.5 删除设备组

功能介绍

应用服务器可调用此接口删除指定设备组。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/device-group/{group_id}

表 1-500 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
group_id	是	String	**参数说明**：设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。 **取值范围**：长度不超过36，十六进制字符串和连接符（-）的组合。

请求参数

表 1-501 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除设备组。

```
DELETE https://{endpoint}/v5/iot/{project_id}/device-group/{group_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```



```
public class DeleteDeviceGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteDeviceGroupRequest request = new DeleteDeviceGroupRequest();
        try {
            DeleteDeviceGroupResponse response = client.deleteDeviceGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteDeviceGroupRequest()
        response = client.delete_device_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteDeviceGroupRequest{}
    response, err := client.DeleteDeviceGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.12.6 管理设备组中的设备

功能介绍

应用服务器可调用此接口管理设备组中的设备。单个设备组内最多添加20,000个设备，一个设备最多可以被添加到10个设备组中。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-group/{group_id}/action

表 1-502 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
group_id	是	String	**参数说明**：设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。 **取值范围**：长度不超过36，十六进制字符串和连接符（-）的组合

表 1-503 Query 参数

参数	是否必选	参数类型	描述
action_id	是	String	**参数说明**：操作类型，支持添加设备和删除设备。 **取值范围**： • addDevice: 添加设备。添加已注册的设备到指定的设备组中。 • removeDevice: 删除设备。从指定的设备组中删除设备，只是解除了设备和设备组的关系，该设备在平台仍然存在。
device_id	是	String	**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。 **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。

请求参数

表 1-504 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

- 管理设备组中的设备，添加设备到设备组。

  ```
  POST https://{endpoint}/v5/iot/{project_id}/device-group/{group_id}/action?
  action_id=addDevice&device_id=yourDeviceId
  ```

- 管理设备组中的设备，从设备组中移除设备。

  ```
  POST https://{endpoint}/v5/iot/{project_id}/device-group/{group_id}/action?
  action_id=removeDevice&device_id=yourDeviceId
  ```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateOrDeleteDeviceInGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateOrDeleteDeviceInGroupRequest request = new CreateOrDeleteDeviceInGroupRequest();
        request.withActionId("<action_id>");
        request.withDeviceId("<device_id>");
        try {
            CreateOrDeleteDeviceInGroupResponse response = client.createOrDeleteDeviceInGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
```

```
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateOrDeleteDeviceInGroupRequest()
        request.action_id = "<action_id>"
        request.device_id = "<device_id>"
        response = client.create_or_delete_device_in_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)
```

```
func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateOrDeleteDeviceInGroupRequest{}
    request.ActionId = "<action_id>"
    request.DeviceId = "<device_id>"
    response, err := client.CreateOrDeleteDeviceInGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.12.7 查询设备组设备列表

功能介绍

应用服务器可调用此接口查询指定设备组下的设备列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-group/{group_id}/devices

表 1-505 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
group_id	是	String	**参数说明**：设备组ID，用于唯一标识一个设备组，在创建设备组时由物联网平台分配。 **取值范围**：长度不超过36，十六进制字符串和连接符（-）的组合。

表 1-506 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-507 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-508 响应 Body 参数

参数	参数类型	描述
devices	Array of **SimplifyDevice** objects	设备列表。
page	**Page** object	查询结果的分页信息。

表 1-509 SimplifyDevice

参数	参数类型	描述
device_id	String	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。
node_id	String	设备标识码，通常使用IMEI、MAC地址或Serial No作为nodeId。
device_name	String	设备名称。

参数	参数类型	描述
product_id	String	设备关联的产品ID，用于唯一标识一个产品模型。

表 1-510 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

查询设备组中设备。

```
POST https://{endpoint}/v5/iot/{project_id}/device-group/{group_id}/devices
```

响应示例

状态码： 200

OK

```
{
  "devices" : [ {
    "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "node_id" : "ABC123456789",
    "device_name" : "dianadevice",
    "product_id" : "b640f4c203b7910fc3cbd446ed437cbd"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```



```
public class ShowDevicesInGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDevicesInGroupRequest request = new ShowDevicesInGroupRequest();
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ShowDevicesInGroupResponse response = client.showDevicesInGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";
```

```
    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDevicesInGroupRequest()
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.show_devices_in_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDevicesInGroupRequest{}
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
```



```
    request.Offset = &offsetRequest
    response, err := client.ShowDevicesInGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.13 标签管理

标签介绍：用户可以给相同属性的一类资源打上相同的标签，便于管理。当前支持标签的资源有Device(设备)。

注意：iot_前缀的为系统预留标签，用户不能使用；一个资源最多只能打10个标签。

1.4.13.1 绑定标签

功能介绍

应用服务器可调用此接口为指定资源绑定标签。当前支持标签的资源有Device(设备)。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/tags/bind-resource

表 1-511 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-512 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-513 请求 Body 参数

参数	是否必选	参数类型	描述
resource_type	是	String	**参数说明**：要绑定标签的资源类型。 **取值范围**： ● device：设备。
resource_id	是	String	**参数说明**：要绑定标签的资源id。例如，资源类型为device，那么对应的资源id就是device_id。 **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。

参数	是否必选	参数类型	描述
tags	是	Array of TagV5DTO objects	**参数说明**：要绑定到指定资源的标签列表，标签列表中各项标签键值之间不允许重复，一个资源最多可以绑定10个标签。

表 1-514 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。
tag_value	否	String	**参数说明**：标签值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。

响应参数

无

请求示例

资源绑定标签，将标签绑定到设备id为d4922d8a的设备上。

```
POST https://{endpoint}/v5/iot/{project_id}/tags/bind-resource

{
  "resource_type" : "device",
  "resource_id" : "d4922d8a",
  "tags" : [ {
    "tag_key" : "testTagName",
    "tag_value" : "testTagValue"
  } ]
}
```

响应示例

无

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.13.2 解绑标签

功能介绍

应用服务器可调用此接口为指定资源解绑标签。当前支持标签的资源有Device(设备)。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/tags/unbind-resource

表 1-515 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-516 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-517 请求 Body 参数

参数	是否必选	参数类型	描述
resource_type	是	String	**参数说明**：要绑定标签的资源类型。 **取值范围**： ● device：设备。
resource_id	是	String	**参数说明**：要绑定标签的资源id。例如，资源类型为device，那么对应的资源id就是device_id。 **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。
tag_keys	是	Array of strings	**参数说明**：指定资源要解绑的标签键列表，标签键列表中各项之间不允许重复，不能填写不存在的标签键值 **取值范围**：标签键长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。

响应参数

无

请求示例

资源解绑标签，将标签testkey从设备id为d4922d8a的设备解绑。

```
POST https://{endpoint}/v5/iot/{project_id}/tags/unbind-resource

{
  "resource_type" : "device",
  "resource_id" : "d4922d8a",
  "tag_keys" : [ "testTag" ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

资源解绑标签，将标签testkey从设备id为d4922d8a的设备解绑。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UntagDeviceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UntagDeviceRequest request = new UntagDeviceRequest();
```



```
        UnbindTagsDTO body = new UnbindTagsDTO();
        List<String> listbodyTagKeys = new ArrayList<>();
        listbodyTagKeys.add("testTag");
        body.withTagKeys(listbodyTagKeys);
        body.withResourceId("d4922d8a");
        body.withResourceType("device");
        request.withBody(body);
        try {
            UntagDeviceResponse response = client.untagDevice(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

资源解绑标签，将标签testkey从设备id为d4922d8a的设备解绑。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UntagDeviceRequest()
        listTagKeysbody = [
            "testTag"
        ]
        request.body = UnbindTagsDTO(
            tag_keys=listTagKeysbody,
            resource_id="d4922d8a",
            resource_type="device"
        )
        response = client.untag_device(request)
```



```
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

资源解绑标签，将标签testkey从设备id为d4922d8a的设备解绑。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UntagDeviceRequest{}
    var listTagKeysbody = []string{
        "testTag",
    }
    request.Body = &model.UnbindTagsDto{
        TagKeys: listTagKeysbody,
        ResourceId: "d4922d8a",
        ResourceType: "device",
    }
    response, err := client.UntagDevice(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.13.3 按标签查询资源

功能介绍

应用服务器可调用此接口查询绑定了指定标签的资源。当前支持标签的资源有Device(设备)。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/tags/query-resources

表 1-518 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-519 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-520 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-521 请求 Body 参数

参数	是否必选	参数类型	描述
resource_type	是	String	**参数说明**：要查询的资源类型，当前支持设备（device）。
tags	是	Array of **TagV5DTO** objects	**参数说明**：标签列表，支持按照标签key和value组合查询，传入的多个标签之间是或的关系。

表 1-522 TagV5DTO

参数	是否必选	参数类型	描述
tag_key	是	String	**参数说明**：标签键，在同一资源下标签键唯一。绑定资源时，如果设置的键已存在，则将覆盖之前的标签值。如果设置的键值不存在，则新增标签。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合。

参数	是否必选	参数类型	描述
tag_value	否	String	**参数说明**：标签值。 **取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。

响应参数

状态码： 200

表 1-523 响应 Body 参数

参数	参数类型	描述
resources	Array of **ResourceDTO** objects	资源列表。
page	**Page** object	查询结果的分页信息。

表 1-524 ResourceDTO

参数	参数类型	描述
resource_id	String	资源id。例如，要查询的资源类型为device，那么对应的资源id就是device_id。

表 1-525 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

按标签查询资源，备查询绑定了指定标签的设备。

```
POST https://{endpoint}/v5/iot/{project_id}/tags/query-resources

{
  "resource_type" : "device",
  "tags" : [ {
    "tag_key" : "testTagName",
    "tag_value" : "testTagValue"
  } ]
}
```

响应示例

状态码： 200

OK

```
{
  "resources" : [ {
    "resource_id" : "d4922d8a"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

按标签查询资源，备查询绑定了指定标签的设备。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class ListResourcesByTagsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListResourcesByTagsRequest request = new ListResourcesByTagsRequest();
        request.withLimit(<limit>);
```



```
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        QueryResourceByTagsDTO body = new QueryResourceByTagsDTO();
        List<TagV5DTO> listbodyTags = new ArrayList<>();
        listbodyTags.add(
            new TagV5DTO()
                .withTagKey("testTagName")
                .withTagValue("testTagValue")
        );
        body.withTags(listbodyTags);
        body.withResourceType("device");
        request.withBody(body);
        try {
            ListResourcesByTagsResponse response = client.listResourcesByTags(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

按标签查询资源，备查询绑定了指定标签的设备。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListResourcesByTagsRequest()
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        listTagsbody = [
```

```
            TagV5DTO(
                tag_key="testTagName",
                tag_value="testTagValue"
            )
        ]
        request.body = QueryResourceByTagsDTO(
            tags=listTagsbody,
            resource_type="device"
        )
        response = client.list_resources_by_tags(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

按标签查询资源，备查询绑定了指定标签的设备。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListResourcesByTagsRequest{}
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    tagValueTags:= "testTagValue"
    var listTagsbody = []model.TagV5Dto{
        {
            TagKey: "testTagName",
```



```
            TagValue: &tagValueTags,
        },
    }
    request.Body = &model.QueryResourceByTagsDto{
        Tags: listTagsbody,
        ResourceType: "device",
    }
    response, err := client.ListResourcesByTags(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.14 资源空间管理

接入凭证是用于AMQP、MQTTS等协议建立长链接时认证使用。

1.4.14.1 查询资源空间列表

功能介绍

资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。应用服务器可以调用此接口查询资源空间列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/apps

表 1-526 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID 最小长度：**0** 最大长度：**32**

表 1-527 Query 参数

参数	是否必选	参数类型	描述
default_app	否	Boolean	**参数说明**：默认资源空间标识，不携带则查询所有资源空间。 **取值范围**： • true：查询默认资源空间。 • false：查询非默认资源空间。

请求参数

表 1-528 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务获取。 最小长度：**0** 最大长度：**1024000**
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**0** 最大长度：**36**

响应参数

状态码： 200

表 1-529 响应 Body 参数

参数	参数类型	描述
applications	Array of ApplicationDTO objects	资源空间信息列表。 数组长度：**0 - 500**

表 1-530 ApplicationDTO

参数	参数类型	描述
app_id	String	资源空间ID，唯一标识一个资源空间，由物联网平台在创建资源空间时分配。资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。 最小长度：**1** 最大长度：**64**
app_name	String	资源空间名称。 最小长度：**1** 最大长度：**64**
create_time	String	资源空间创建时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。 最小长度：**1** 最大长度：**64**
default_app	Boolean	是否为默认资源空间

请求示例

列表查询资源。

```
GET https://{endpoint}/v5/iot/{project_id}/apps
```

响应示例

状态码： 200

Successful response

```
{
  "applications" : [ {
    "app_id" : "0ab87ceecbfc49acbcc8d5acdef3c68c",
    "app_name" : "testApp",
    "create_time" : "20151212T121212Z",
```

```
    "default_app" : true
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowApplicationsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowApplicationsRequest request = new ShowApplicationsRequest();
        request.withDefaultApp(<default_app>);
        try {
            ShowApplicationsResponse response = client.showApplications(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowApplicationsRequest()
        request.default_app = <DefaultApp>
        response = client.show_applications(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
```

```
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowApplicationsRequest{}
    defaultAppRequest:= <default_app>
    request.DefaultApp = &defaultAppRequest
    response, err := client.ShowApplications(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Successful response
400	Bad Request
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.14.2 创建资源空间

功能介绍

资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。应用服务器可以调用此接口创建资源空间。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/apps

表 1-531 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID 最小长度：**0** 最大长度：**32**

请求参数

表 1-532 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务获取。 最小长度：**0** 最大长度：**1024000**
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**0** 最大长度：**36**

表 1-533 请求 Body 参数

参数	是否必选	参数类型	描述
app_name	是	String	**参数说明**：资源空间名称。 **取值范围**：长度不超过64，允许中文、字母、数字、下划线（_）、连接符（-）等一些特殊字符的组合。 最小长度：**1** 最大长度：**64**

响应参数

状态码： 201

表 1-534 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID，唯一标识一个资源空间，由物联网平台在创建资源空间时分配。资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。 最小长度：**1** 最大长度：**64**
app_name	String	资源空间名称。 最小长度：**1** 最大长度：**64**
create_time	String	资源空间创建时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。 最小长度：**1** 最大长度：**64**
default_app	Boolean	是否为默认资源空间

请求示例

创建资源空间，资源空间名为testapp。

```
POST https://{endpoint}/v5/iot/{project_id}/apps

{
  "app_name" : "testApp"
}
```

响应示例

状态码： 201

Created

```
{
  "app_id" : "0ab87ceecbfc49acbcc8d5acdef3c68c",
  "app_name" : "testApp",
  "create_time" : "20151212T121212Z",
  "default_app" : true
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建资源空间，资源空间名为testapp。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class AddApplicationSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddApplicationRequest request = new AddApplicationRequest();
        AddApplication body = new AddApplication();
        body.withAppName("testApp");
        request.withBody(body);
        try {
            AddApplicationResponse response = client.addApplication(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建资源空间，资源空间名为testapp。


```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddApplicationRequest()
        request.body = AddApplication(
            app_name="testApp"
        )
        response = client.add_application(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建资源空间，资源空间名为testapp。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"
```

```
    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddApplicationRequest{}
    request.Body = &model.AddApplication{
        AppName: "testApp",
    }
    response, err := client.AddApplication(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.14.3 查询资源空间

功能介绍

资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。应用服务器可以调用此接口查询指定资源空间详情。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/apps/{app_id}

表 1-535 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID 最小长度：**0** 最大长度：**32**
app_id	是	String	**参数说明**：资源空间ID，唯一标识一个资源空间，由物联网平台在创建资源空间时分配。资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-536 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务获取。 最小长度：**0** 最大长度：**1024000**
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**0** 最大长度：**36**

响应参数

状态码： 200

表 1-537 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID，唯一标识一个资源空间，由物联网平台在创建资源空间时分配。资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。 最小长度：**1** 最大长度：**64**
app_name	String	资源空间名称。 最小长度：**1** 最大长度：**64**
create_time	String	资源空间创建时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。 最小长度：**1** 最大长度：**64**
default_app	Boolean	是否为默认资源空间

请求示例

查询资源空间详情。

```
GET https://{endpoint}/v5/iot/{project_id}/apps/{app_id}
```

响应示例

状态码： 200

OK

```
{
  "app_id" : "0ab87ceecbfc49acbcc8d5acdef3c68c",
  "app_name" : "testApp",
  "create_time" : "20151212T121212Z",
  "default_app" : true
}
```

状态码

状态码	描述
200	OK
400	Bad Request

状态码	描述
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.14.4 删除资源空间

功能介绍

删除指定资源空间。删除资源空间属于高危操作，删除资源空间后，该空间下的产品、设备等资源将不可用，请谨慎操作！

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/apps/{app_id}

表 1-538 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID 最小长度：**0** 最大长度：**32**
app_id	是	String	**参数说明**：资源空间ID，唯一标识一个资源空间，由物联网平台在创建资源空间时分配。资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-539 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务获取。 最小长度：**0** 最大长度：**1024000**
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**0** 最大长度：**36**

响应参数

无

请求示例

删除资源空间。

```
DELETE https://{endpoint}/v5/iot/{project_id}/apps/{app_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```

```
public class DeleteApplicationSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteApplicationRequest request = new DeleteApplicationRequest();
        try {
            DeleteApplicationResponse response = client.deleteApplication(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteApplicationRequest()
        response = client.delete_application(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteApplicationRequest{}
    response, err := client.DeleteApplication(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.14.5 更新资源空间

功能介绍

应用服务器可以调用此接口更新资源空间的名称

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/apps/{app_id}

表 1-540 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID 最小长度：**0** 最大长度：**32**

参数	是否必选	参数类型	描述
app_id	是	String	**参数说明**：资源空间ID，唯一标识一个资源空间，由物联网平台在创建资源空间时分配。资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

请求参数

表 1-541 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务获取。 最小长度：**0** 最大长度：**1024000**
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**0** 最大长度：**36**

表 1-542 请求 Body 参数

参数	是否必选	参数类型	描述
app_name	否	String	**参数说明**：资源空间名称。 **取值范围**：长度不超过64，允许中文、字母、数字、下划线（_）、连接符（-）等一些特殊字符的组合。 最小长度：**1** 最大长度：**64**

响应参数

状态码： 200

表 1-543 响应 Body 参数

参数	参数类型	描述
app_id	String	资源空间ID，唯一标识一个资源空间，由物联网平台在创建资源空间时分配。资源空间对应的是物联网平台原有的应用，在物联网平台的含义与应用一致，只是变更了名称。 最小长度：**1** 最大长度：**64**
app_name	String	资源空间名称。 最小长度：**1** 最大长度：**64**
create_time	String	资源空间创建时间，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。 最小长度：**1** 最大长度：**64**
default_app	Boolean	是否为默认资源空间

请求示例

```
PUT https://{endpoint}/v5/iot/{project_id}/apps/{app_id}

{
  "app_name" : "testApp"
}
```

响应示例

状态码： 200

OK

```
{
  "app_id" : "0ab87ceecbfc49acbcc8d5acdef3c68c",
  "app_name" : "testApp",
  "create_time" : "20151212T121212Z",
  "default_app" : true
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;
```

```
import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateApplicationSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateApplicationRequest request = new UpdateApplicationRequest();
        UpdateApplicationDTO body = new UpdateApplicationDTO();
        body.withAppName("testApp");
        request.withBody(body);
        try {
            UpdateApplicationResponse response = client.updateApplication(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *
```

```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateApplicationRequest()
        request.body = UpdateApplicationDTO(
            app_name="testApp"
        )
        response = client.update_application(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
```

```
        WithRegion(region.NewRegion("cn-north-4", endpoint)).
        WithCredential(auth).
        Build())

    request := &model.UpdateApplicationRequest{}
    appNameUpdateApplicationDto:= "testApp"
    request.Body = &model.UpdateApplicationDto{
        AppName: &appNameUpdateApplicationDto,
    }
    response, err := client.UpdateApplication(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15 批量任务

批量任务场景下，用户通过调用此目录接口实现任务文件管理功能。

- 当前支持批量创建设备任务、批量删除设备任务、批量冻结设备任务、批量解冻设备任务的文件管理。

- 当前单用户管理的文件数量最大为10，超过则无法上传新的文件。

- 当前文件最大存储时间为1小时，超过1小时文件会被平台自动老化。

1.4.15.1 创建批量任务

功能介绍

应用服务器可调用此接口为创建批量处理任务，对多个设备进行批量操作。当前支持批量软固件升级、批量创建设备、批量修改设备、批量删除设备、批量冻结设备、批量解冻设备、批量创建命令、批量创建消息任务。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/batchtasks

表 1-544 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-545 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

表 1-546 请求 Body 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的批量任务归属到哪个资源空间下，否则创建的批量任务将会归属到**默认资源空间**下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
task_name	是	String	**参数说明**：批量任务名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、下划线（_）的组合。 最小长度：**1** 最大长度：**128**
task_type	是	String	**参数说明**：批量任务类型。 **取值范围**： • softwareUpgrade: 软件升级任务 • firmwareUpgrade: 固件升级任务 • createDevices: 批量创建设备任务 • deleteDevices: 批量删除设备任务 • freezeDevices: 批量冻结设备任务 • unfreezeDevices: 批量解冻设备任务 • createCommands: 批量创建同步命令任务 • createAsyncCommands: 批量创建异步命令任务 • createMessages: 批量创建消息任务 • updateDeviceShadows：批量配置设备影子任务 • updateDevices：批量更新设备任务

参数	是否必选	参数类型	描述
task_mode	否	String	**参数说明**：批量任务的模式，当前只支持网关模式，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，若升级的设备为某个网关的子设备，则平台下发获取版本信息通知和平台下发升级通知将携带task_id（软固件升级批量任务的任务ID）和sub_device_count（批量任务中网关设备包含的升级子设备数量）字段。 **取值范围**：GATEWAY: 网关模式。
task_ext_info	否	Object	**参数说明**：批量任务额外扩展信息，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，平台下发获取版本信息通知和平台下发升级通知将携带该字段。 **取值范围**：最长不超过512个字符。 最大长度：**512**
targets	否	Array of strings	**参数说明**：执行批量任务的目标，此处填写device_id列表，且最多支持3万个device_id。当task_type为firmwareUpgrade，softwareUpgrade，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows，支持该参数。同时使用targets、targets_filter、document_source参数时，只有一个参数会生效，且平台优先使用targets，其次是targets_filter，最后是document_source。 **取值范围**：device_id列表。device_id支持长度不超过128，只允许字母、数字、下划线（_）、连接符（-）的组合。

参数	是否必选	参数类型	描述
targets_filter	否	Map<String,Object>	**参数说明**：任务目标筛选参数。Json格式，里面是一个个键值对，（K,V）格式标识筛选targets需要的参数，目前支持的K有group_ids（V填写group_id数组，eg:["e495cf17-ff79-4294-8f64-4d367919d665"]，任务则会筛选出来符合该群组条件的设备作为目标）。当task_type为firmwareUpgrade，softwareUpgrade，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows，支持该参数。同时使用targets、targets_filter、document_source参数时，只有一个参数会生效，且平台优先使用targets，其次是targets_filter，最后是document_source。

参数	是否必选	参数类型	描述
document	否	Object	**参数说明**：执行任务数据文档，Json格式，Json里面是(K,V)键值对。当task_type为firmwareUpgrade，softwareUpgrade，createCommands，createAsyncCommands，createMessages，updateDeviceShadows，支持该参数。 • softwareUpgrade| firmwareUpgrade，需要填写key为package_id，value为在平台上传的软固件附件id，id由portal软件库包管理上传并查询获得。eg：“{"package_id": "32822e5744a45ede319d2c50"}”。 • createCommands，需要填写同步命令相关参数，eg：“{"service_id":"water","command_name":"ON_OFF","paras":{"value":"ON"}}”，参考**设备同步命令**)。 • createAsyncCommands，需要填写异步命令相关参数，eg：“{"service_id":"water","command_name":"ON_OFF","paras": {"value":"ON"},"expire_time":0,"send_strategy":"immediately"}”，参考**设备异步命令**)。 • createMessages，需要填写消息下发相关参数，eg：“{"message_id":"99b32da9-cd17-4cdf-a286-f6e849cbc364","name":"messageName","message":"HelloWorld","encoding":"none","payload_format":"standard","topic":"messageDown","topic_full_name":"/device/message/down"}”，参考**下发设备消息**)。 • updateDeviceShadows，需要填写配置设备影子相关参

参数	是否必选	参数类型	描述
			数，eg：“{"shadow": [{"service_id": "WaterMeter","desired": {"temperature": "60"}}]}”，参考**配置设备影子预期数据**)。限制说明：1. service_id和desired参数必填。2. 配置的service_id数量不大于5个且service_id不能重复。3. 配置参数内容大小不超过10K。
task_policy	否	**TaskPolicy** object	**参数说明**：任务策略请求信息。
document_source	否	String	**参数说明**：上传的批量任务文件ID。当task_type为createDevices，updateDevices，deleteDevices，freezeDevices，unfreezeDevices时，支持该参数。使用该参数时，需要先调用批量任务的文件管理接口上传文件来获取文件ID，文件样例请参见 **批量注册设备模板**，**批量更新设备模板**，**批量删除设备模板**，**批量冻结设备模板**，**批量解冻设备模板**。同时使用targets、targets_filter、document_source参数时，只有一个参数会生效，且平台优先使用targets，其次是targets_filter，最后是document_source。

表 1-547 TaskPolicy

参数	是否必选	参数类型	描述
schedule_time	否	String	**参数说明**：批量任务指定执行时间。 **取值范围**：7天内，不传入此参数表示立即执行，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

参数	是否必选	参数类型	描述
retry_count	否	Integer	**参数说明**：批量任务子任务自动重试次数。 **取值范围**：如果传入retry_interval参数，则需传入该参数，最大支持重试5次。 最小值：**1** 最大值：**5**
retry_interval	否	Integer	**参数说明**：批量任务子任务失败后，自动重试时间间隔，单位：分钟。 **取值范围**：最大1440(24小时)，不传入此参数表示不重试，如果传入retry_count参数则需要传入该参数。 最小值：**0** 最大值：**1440**

响应参数

状态码： 201

表 1-548 响应 Body 参数

参数	参数类型	描述
task_id	String	批量任务ID，创建批量任务时由物联网平台分配获得。
task_name	String	批量任务名称。

参数	参数类型	描述
task_type	String	批量任务类型，取值范围：firmwareUpgrade，softwareUpgrade，createDevices，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows。 ● softwareUpgrade: 软件升级任务 ● firmwareUpgrade: 固件升级任务 ● createDevices: 批量创建设备任务 ● deleteDevices: 批量删除设备任务 ● freezeDevices: 批量冻结设备任务 ● unfreezeDevices: 批量解冻设备任务 ● createCommands: 批量创建同步命令任务 ● createAsyncCommands: 批量创建异步命令任务 ● createMessages: 批量创建消息任务 ● updateDeviceShadows: 批量配置设备影子任务 ● updateDevices：批量更新设备任务
task_mode	String	**参数说明**：批量任务的模式，当前只支持网关模式，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，若升级的设备为某个网关的子设备，则平台下发获取版本信息通知和平台下发升级通知将携带task_id（软固件升级批量任务的任务ID）和sub_device_count（批量任务中网关设备包含的升级子设备数量）字段。 **取值范围**：GATEWAY: 网关模式。
task_ext_info	Object	**参数说明**：批量任务额外扩展信息，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，平台下发获取版本信息通知和平台下发升级通知将携带该字段。 **取值范围**：最长不超过512个字符。 最大长度：**512**
targets	Array of strings	执行批量任务的目标，当task_type为firmwareUpgrade，softwareUpgrade，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows，此处填写device_id列表。

参数	参数类型	描述
targets_filter	Map<String,Object>	任务目标筛选参数。Json格式，里面是一个个键值对，（K,V）格式标识筛选targets需要的参数，目前支持的K有group_ids（V填写group_id数组，eg:["e495cf17-ff79-4294-8f64-4d367919d665"]，任务则会筛选出来符合该群组条件的设备作为目标）
document	Object	执行任务数据文档，Json格式。(当task_type为softwareUpgrade|firmwareUpgrade，也就是软固件升级任务时，Json里面是(K,V)键值对，需要填写key为package_id，value为在平台上传的软固件附件id，id由portal软件库包管理上传并查询获得。当task_type为createCommands，也就是批量创建同步命令任务时，Json里面是命令相关参数，eg：{"service_id":"water","command_name":"ON_OFF","paras":{"value":"ON"}}，参考**设备同步命令**)。当task_type为createAsyncCommands，也就是批量创建异步命令任务时，Json里面是命令相关参数，eg：{"service_id":"water","command_name":"ON_OFF","paras":{"value":"ON"},"expire_time":0,"send_strategy":"immediately"}，参考**设备异步命令**)。当task_type为updateDeviceShadows，也就是批量配置设备影子任务时，Json里面是命令相关参数，eg：{"shadow": [{"service_id":"WaterMeter","desired": {"temperature":"60"}}]}，参考**配置设备影子预期数据**)。
task_policy	**TaskPolicy** object	任务执行策略。
status	String	批量任务的状态，可选参数，取值范围：Success|Fail|Processing|PartialSuccess|Stopped|Waitting|Initializing|Stopping。 • Initializing: 初始化中。 • Waitting: 等待中。 • Processing: 执行中。 • Success: 成功。 • Fail: 失败。 • PartialSuccess: 部分成功。 • Stopped: 停止。 • Stopping 停止中。
status_desc	String	批量任务状态描述(包含主任务失败错误信息)
task_progress	**TaskProgress** object	子任务执行统计结果。

参数	参数类型	描述
create_time	String	批量任务的创建时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-549 TaskPolicy

参数	参数类型	描述
schedule_time	String	**参数说明**：批量任务指定执行时间。 **取值范围**：7天内，不传入此参数表示立即执行，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
retry_count	Integer	**参数说明**：批量任务子任务自动重试次数。 **取值范围**：如果传入retry_interval参数，则需传入该参数，最大支持重试5次。 最小值：**1** 最大值：**5**
retry_interval	Integer	**参数说明**：批量任务子任务失败后，自动重试时间间隔，单位：分钟。 **取值范围**：最大1440(24小时)，不传入此参数表示不重试，如果传入retry_count参数则需要传入该参数。 最小值：**0** 最大值：**1440**

表 1-550 TaskProgress

参数	参数类型	描述
total	Integer	子任务总个数。
processing	Integer	正在执行的子任务个数。
success	Integer	执行成功的子任务个数。
fail	Integer	执行失败的的子任务个数。
waitting	Integer	等待执行的子任务个数。
fail_wait_retry	Integer	失败等待重试的子任务个数。
stopped	Integer	停止的子任务个数。
removed	Integer	移除的子任务个数。

请求示例

- 创建批量软件升级任务，指定设备升级。

```
POST https://{endpoint}/v5/iot/{project_id}/batchtasks

{
  "app_id" : "Ev8FVvCfOdQDzrFrxSOemiw_aMca",
  "task_name" : "BatchSoftwareUpgradeTask",
  "task_type" : "softwareUpgrade",
  "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ],
  "document" : {
    "package_id" : "32822e5744a45ede319d2c50"
  },
  "task_policy" : {
    "schedule_time" : "20151212T121212Z",
    "retry_count" : 5,
    "retry_interval" : 60
  }
}
```

- 创建批量软件升级任务，指定设备组中设备进行升级。

```
POST https://{endpoint}/v5/iot/{project_id}/batchtasks

{
  "app_id" : "Ev8FVvCfOdQDzrFrxSOemiw_aMca",
  "task_name" : "BatchSoftwareUpgradeTask",
  "task_type" : "softwareUpgrade",
  "document" : {
    "package_id" : "32822e5744a45ede319d2c50"
  },
  "targets_filter" : {
    "group_ids" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
  },
  "task_policy" : {
    "schedule_time" : "20151212T121212Z",
    "retry_count" : 5,
    "retry_interval" : 60
  }
}
```

- 创建批量注册设备任务，从文件中读取参数。

```
POST https://{endpoint}/v5/iot/{project_id}/batchtasks

{
  "app_id" : "Ev8FVvCfOdQDzrFrxSOemiw_aMca",
  "task_name" : "BatchCreateDevicesTask",
  "task_type" : "createDevices",
  "task_policy" : {
    "schedule_time" : "20151212T121212Z",
    "retry_count" : 5,
    "retry_interval" : 60
  },
  "document_source" : "jeQDJQZltU8iKgFFoW060F5SGZka"
}
```

响应示例

状态码： 201

Created

```
{
  "task_id" : "5c8ba99030344005c02316ad",
  "task_name" : "testname",
  "task_type" : "softwareUpgrade",
  "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ],
  "targets_filter" : {
```



```
    "group_ids" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
  },
  "document" : {
    "package_id" : "32822e5744a45ede319d2c50"
  },
  "task_policy" : {
    "schedule_time" : "20151212T121212Z",
    "retry_count" : 5,
    "retry_interval" : 60
  },
  "status" : "Success",
  "status_desc" : "string",
  "task_progress" : {
    "total" : 0,
    "processing" : 0,
    "success" : 0,
    "fail" : 0,
    "waitting" : 0,
    "fail_wait_retry" : 0,
    "stopped" : 0
  },
  "create_time" : "20151212T121212Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建批量软件升级任务，指定设备升级。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);
```



```
        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateBatchTaskRequest request = new CreateBatchTaskRequest();
        CreateBatchTask body = new CreateBatchTask();
        TaskPolicy taskPolicybody = new TaskPolicy();
        taskPolicybody.withScheduleTime("20151212T121212Z")
            .withRetryCount(5)
            .withRetryInterval(60);
        List<String> listbodyTargets = new ArrayList<>();
        listbodyTargets.add("e495cf17-ff79-4294-8f64-4d367919d665");
        body.withTaskPolicy(taskPolicybody);
        body.withDocument("{\"package_id\":\"32822e5744a45ede319d2c50\"}");
        body.withTargets(listbodyTargets);
        body.withTaskType("softwareUpgrade");
        body.withTaskName("BatchSoftwareUpgradeTask");
        body.withAppId("Ev8FVvCfOdQDzrFrxSOemiw_aMca");
        request.withBody(body);
        try {
            CreateBatchTaskResponse response = client.createBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建批量软件升级任务，指定设备组中设备进行升级。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.Map;
import java.util.HashMap;

public class CreateBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
```



```
            .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
            .withAk(ak)
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        CreateBatchTaskRequest request = new CreateBatchTaskRequest();
        CreateBatchTask body = new CreateBatchTask();
        TaskPolicy taskPolicybody = new TaskPolicy();
        taskPolicybody.withScheduleTime("20151212T121212Z")
            .withRetryCount(5)
            .withRetryInterval(60);
        Map<String, Object> listbodyTargetsFilter = new HashMap<>();
        listbodyTargetsFilter.put("group_ids", "[e495cf17-ff79-4294-8f64-4d367919d665]");
        body.withTaskPolicy(taskPolicybody);
        body.withDocument("{\"package_id\":\"32822e5744a45ede319d2c50\"}");
        body.withTargetsFilter(listbodyTargetsFilter);
        body.withTaskType("softwareUpgrade");
        body.withTaskName("BatchSoftwareUpgradeTask");
        body.withAppId("Ev8FVvCfOdQDzrFrxSOemiw_aMca");
        request.withBody(body);
        try {
            CreateBatchTaskResponse response = client.createBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建批量注册设备任务，从文件中读取参数。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";
```

```
        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateBatchTaskRequest request = new CreateBatchTaskRequest();
        CreateBatchTask body = new CreateBatchTask();
        TaskPolicy taskPolicybody = new TaskPolicy();
        taskPolicybody.withScheduleTime("20151212T121212Z")
            .withRetryCount(5)
            .withRetryInterval(60);
        body.withDocumentSource("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withTaskPolicy(taskPolicybody);
        body.withTaskType("createDevices");
        body.withTaskName("BatchCreateDevicesTask");
        body.withAppId("Ev8FVvCfOdQDzrFrxSOemiw_aMca");
        request.withBody(body);
        try {
            CreateBatchTaskResponse response = client.createBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建批量软件升级任务，指定设备升级。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```



```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateBatchTaskRequest()
        taskPolicybody = TaskPolicy(
            schedule_time="20151212T121212Z",
            retry_count=5,
            retry_interval=60
        )
        listTargetsbody = [
            "e495cf17-ff79-4294-8f64-4d367919d665"
        ]
        request.body = CreateBatchTask(
            task_policy=taskPolicybody,
            document="{\"package_id\":\"32822e5744a45ede319d2c50\"}",
            targets=listTargetsbody,
            task_type="softwareUpgrade",
            task_name="BatchSoftwareUpgradeTask",
            app_id="Ev8FVvCfOdQDzrFrxSOemiw_aMca"
        )
        response = client.create_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建批量软件升级任务，指定设备组中设备进行升级。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateBatchTaskRequest()
        taskPolicybody = TaskPolicy(
            schedule_time="20151212T121212Z",
```

```
            retry_count=5,
            retry_interval=60
        )
        listTargetsFilterbody = {
            "group_ids": "[e495cf17-ff79-4294-8f64-4d367919d665]"
        }
        request.body = CreateBatchTask(
            task_policy=taskPolicybody,
            document="{\"package_id\":\"32822e5744a45ede319d2c50\"}",
            targets_filter=listTargetsFilterbody,
            task_type="softwareUpgrade",
            task_name="BatchSoftwareUpgradeTask",
            app_id="Ev8FVvCfOdQDzrFrxSOemiw_aMca"
        )
        response = client.create_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建批量注册设备任务，从文件中读取参数。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateBatchTaskRequest()
        taskPolicybody = TaskPolicy(
            schedule_time="20151212T121212Z",
            retry_count=5,
            retry_interval=60
        )
        request.body = CreateBatchTask(
            document_source="jeQDJQZltU8iKgFFoW060F5SGZka",
            task_policy=taskPolicybody,
            task_type="createDevices",
            task_name="BatchCreateDevicesTask",
            app_id="Ev8FVvCfOdQDzrFrxSOemiw_aMca"
        )
        response = client.create_batch_task(request)
        print(response)
```

```
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建批量软件升级任务，指定设备升级。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateBatchTaskRequest{}
    scheduleTimeTaskPolicy:= "20151212T121212Z"
    retryCountTaskPolicy:= int32(5)
    retryIntervalTaskPolicy:= int32(60)
    taskPolicybody := &model.TaskPolicy{
        ScheduleTime: &scheduleTimeTaskPolicy,
        RetryCount: &retryCountTaskPolicy,
        RetryInterval: &retryIntervalTaskPolicy,
    }
    var listTargetsbody = []string{
        "e495cf17-ff79-4294-8f64-4d367919d665",
    }
    var documentCreateBatchTask interface{} = "{\"package_id\":\"32822e5744a45ede319d2c50\"}"
    appIdCreateBatchTask:= "Ev8FVvCfOdQDzrFrxSOemiw_aMca"
    request.Body = &model.CreateBatchTask{
        TaskPolicy: taskPolicybody,
        Document: &documentCreateBatchTask,
        Targets: &listTargetsbody,
        TaskType: "softwareUpgrade",
        TaskName: "BatchSoftwareUpgradeTask",
        AppId: &appIdCreateBatchTask,
    }
```



```
    response, err := client.CreateBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建批量软件升级任务，指定设备组中设备进行升级。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateBatchTaskRequest{}
    scheduleTimeTaskPolicy:= "20151212T121212Z"
    retryCountTaskPolicy:= int32(5)
    retryIntervalTaskPolicy:= int32(60)
    taskPolicybody := &model.TaskPolicy{
        ScheduleTime: &scheduleTimeTaskPolicy,
        RetryCount: &retryCountTaskPolicy,
        RetryInterval: &retryIntervalTaskPolicy,
    }
    var listTargetsFilterbody = map[string]interface{}{
        "group_ids": "[e495cf17-ff79-4294-8f64-4d367919d665]",
    }
    var documentCreateBatchTask interface{} = "{\"package_id\":\"32822e5744a45ede319d2c50\"}"
    appIdCreateBatchTask:= "Ev8FVvCfOdQDzrFrxSOemiw_aMca"
    request.Body = &model.CreateBatchTask{
        TaskPolicy: taskPolicybody,
        Document: &documentCreateBatchTask,
        TargetsFilter: listTargetsFilterbody,
        TaskType: "softwareUpgrade",
        TaskName: "BatchSoftwareUpgradeTask",
        AppId: &appIdCreateBatchTask,
    }
    response, err := client.CreateBatchTask(request)
```

```
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建批量注册设备任务，从文件中读取参数。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateBatchTaskRequest{}
    scheduleTimeTaskPolicy:= "20151212T121212Z"
    retryCountTaskPolicy:= int32(5)
    retryIntervalTaskPolicy:= int32(60)
    taskPolicybody := &model.TaskPolicy{
        ScheduleTime: &scheduleTimeTaskPolicy,
        RetryCount: &retryCountTaskPolicy,
        RetryInterval: &retryIntervalTaskPolicy,
    }
    documentSourceCreateBatchTask:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    appIdCreateBatchTask:= "Ev8FVvCfOdQDzrFrxSOemiw_aMca"
    request.Body = &model.CreateBatchTask{
        DocumentSource: &documentSourceCreateBatchTask,
        TaskPolicy: taskPolicybody,
        TaskType: "createDevices",
        TaskName: "BatchCreateDevicesTask",
        AppId: &appIdCreateBatchTask,
    }
    response, err := client.CreateBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```

```
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	BAD REQUEST
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.2 查询批量任务列表

功能介绍

应用服务器可调用此接口查询物联网平台中批量任务列表，每一个任务又包括具体的任务内容、任务状态、任务完成情况统计等。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/batchtasks

表 1-551 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-552 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的任务列表，不携带该参数则会查询该用户下所有任务列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
task_type	是	String	**参数说明**：批量任务类型。 **取值范围**： ● softwareUpgrade: 软件升级任务 ● firmwareUpgrade: 固件升级任务 ● createDevices: 批量创建设备任务 ● deleteDevices: 批量删除设备任务 ● freezeDevices: 批量冻结设备任务 ● unfreezeDevices: 批量解冻设备任务 ● createCommands: 批量创建同步命令任务 ● createAsyncCommands: 批量创建异步命令任务 ● createMessages: 批量创建消息任务 ● updateDeviceShadows：批量配置设备影子任务 ● updateDevices：批量更新设备任务

参数	是否必选	参数类型	描述
status	否	String	**参数说明**：批量任务的状态，可选参数。 **取值范围**： ● Initializing: 初始化中。 ● Waitting: 等待中。 ● Processing: 执行中。 ● Success: 成功。 ● Fail: 失败。 ● PartialSuccess: 部分成功。 ● Stopped: 停止。 ● Stopping: 停止中。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-553 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

响应参数

状态码： 200

表 1-554 响应 Body 参数

参数	参数类型	描述
batchtasks	Array of **Task** objects	批量任务列表。
page	**Page** object	查询结果的分页信息。

表 1-555 Task

参数	参数类型	描述
task_id	String	批量任务ID，创建批量任务时由物联网平台分配获得。
task_name	String	批量任务名称。
task_type	String	批量任务类型，取值范围：firmwareUpgrade，softwareUpgrade，createDevices，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows。 ● softwareUpgrade: 软件升级任务 ● firmwareUpgrade: 固件升级任务 ● createDevices: 批量创建设备任务 ● deleteDevices: 批量删除设备任务 ● freezeDevices: 批量冻结设备任务 ● unfreezeDevices: 批量解冻设备任务 ● createCommands: 批量创建同步命令任务 ● createAsyncCommands: 批量创建异步命令任务 ● createMessages: 批量创建消息任务 ● updateDeviceShadows: 批量配置设备影子任务 ● updateDevices：批量更新设备任务

参数	参数类型	描述
task_mode	String	**参数说明**：批量任务的模式，当前只支持网关模式，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，若升级的设备为某个网关的子设备，则平台下发获取版本信息通知和平台下发升级通知将携带task_id（软固件升级批量任务的任务ID）和sub_device_count（批量任务中网关设备包含的升级子设备数量）字段。 **取值范围**：GATEWAY:网关模式。
task_ext_info	Object	**参数说明**：批量任务额外扩展信息，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，平台下发获取版本信息通知和平台下发升级通知将携带该字段。 **取值范围**：最长不超过512个字符。 最大长度：**512**
targets	Array of strings	执行批量任务的目标，当task_type为firmwareUpgrade，softwareUpgrade，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows，此处填写device_id列表。
targets_filter	Map<String,Object>	任务目标筛选参数。Json格式，里面是一个个键值对，（K,V）格式标识筛选targets需要的参数，目前支持的K有group_ids（V填写group_id数组，eg:["e495cf17-ff79-4294-8f64-4d367919d665"]，任务则会筛选出来符合该群组条件的设备作为目标）

参数	参数类型	描述
document	Object	执行任务数据文档，Json格式。(当task_type为softwareUpgrade|firmwareUpgrade，也就是软固件升级任务时，Json里面是(K,V)键值对，需要填写key为package_id，value为在平台上传的软固件附件id，id由portal软件库包管理上传并查询获得。当task_type为createCommands，也就是批量创建同步命令任务时，Json里面是命令相关参数，eg：{"service_id":"water","command_name":"ON_OFF","paras":{"value":"ON"}}，参考**设备同步命令**)。当task_type为createAsyncCommands，也就是批量创建异步命令任务时，Json里面是命令相关参数，eg：{"service_id":"water","command_name":"ON_OFF","paras":{"value":"ON"},"expire_time":0,"send_strategy":"immediately"}，参考**设备异步命令**)。当task_type为updateDeviceShadows，也就是批量配置设备影子任务时，Json里面是命令相关参数，eg：{"shadow": [{"service_id":"WaterMeter","desired": {"temperature":"60"}}]}，参考**配置设备影子预期数据**)。
task_policy	**TaskPolicy** object	任务执行策略。
status	String	批量任务的状态，可选参数，取值范围：Success|Fail|Processing|PartialSuccess|Stopped|Waitting|Initializing|Stopping。 • Initializing: 初始化中。 • Waitting: 等待中。 • Processing: 执行中。 • Success: 成功。 • Fail: 失败。 • PartialSuccess: 部分成功。 • Stopped: 停止。 • Stopping 停止中。
status_desc	String	批量任务状态描述(包含主任务失败错误信息)
task_progress	**TaskProgress** object	子任务执行统计结果。
create_time	String	批量任务的创建时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-556 TaskPolicy

参数	参数类型	描述
schedule_time	String	**参数说明**：批量任务指定执行时间。 **取值范围**：7天内，不传入此参数表示立即执行，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
retry_count	Integer	**参数说明**：批量任务子任务自动重试次数。 **取值范围**：如果传入retry_interval参数，则需传入该参数，最大支持重试5次。 最小值：**1** 最大值：**5**
retry_interval	Integer	**参数说明**：批量任务子任务失败后，自动重试时间间隔，单位：分钟。 **取值范围**：最大1440(24小时)，不传入此参数表示不重试，如果传入retry_count参数则需要传入该参数。 最小值：**0** 最大值：**1440**

表 1-557 TaskProgress

参数	参数类型	描述
total	Integer	子任务总个数。
processing	Integer	正在执行的子任务个数。
success	Integer	执行成功的子任务个数。
fail	Integer	执行失败的的子任务个数。
waitting	Integer	等待执行的子任务个数。
fail_wait_retry	Integer	失败等待重试的子任务个数。
stopped	Integer	停止的子任务个数。
removed	Integer	移除的子任务个数。

表 1-558 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

列表查询出所有批量任务。

```
GET https://{endpoint}/v5/iot/{project_id}/batchtasks
```

响应示例

状态码： 200

OK

```
{
  "batchtasks" : [ {
    "task_id" : "5c8ba99030344005c02316ad",
    "task_name" : "testname",
    "task_type" : "softwareUpgrade",
    "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ],
    "targets_filter" : {
      "group_ids" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
    },
    "document" : {
      "package_id" : "32822e5744a45ede319d2c50"
    },
    "task_policy" : {
      "schedule_time" : "20151212T121212Z",
      "retry_count" : 5,
      "retry_interval" : 60
    },
    "status" : "Success",
    "status_desc" : "string",
    "task_progress" : {
      "total" : 0,
      "processing" : 0,
      "success" : 0,
      "fail" : 0,
      "waitting" : 0,
      "fail_wait_retry" : 0,
      "stopped" : 0
    },
    "create_time" : "20151212T121212Z"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```



```
public class ListBatchTasksSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListBatchTasksRequest request = new ListBatchTasksRequest();
        request.withAppId("<app_id>");
        request.withTaskType("<task_type>");
        request.withStatus("<status>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListBatchTasksResponse response = client.listBatchTasks(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
```

```
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListBatchTasksRequest()
        request.app_id = "<app_id>"
        request.task_type = "<task_type>"
        request.status = "<status>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_batch_tasks(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())
```

```
    request := &model.ListBatchTasksRequest{}
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    request.TaskType = "<task_type>"
    statusRequest:= "<status>"
    request.Status = &statusRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListBatchTasks(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.3 查询批量任务

功能介绍

应用服务器可调用此接口查询物联网平台中指定批量任务的信息，包括任务内容、任务状态、任务完成情况统计以及子任务列表等。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/batchtasks/{task_id}

表 1-559 路径参数

参数	是否必选	参数类型	描述
task_id	是	String	**参数说明**：批量任务ID，创建批量任务时由物联网平台分配获得。 **取值范围**：长度不超过24，只允许小写字母a到f、数字的组合。
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-560 Query 参数

参数	是否必选	参数类型	描述
task_detail_status	否	String	**参数说明**：子任务的执行状态，可选参数。 **取值范围**： • Success: 成功。 • Fail: 失败。 • Processing: 执行中。 • FailWaitRetry: 失败重试。 • Stopped: 已停止。 • Waitting: 等待执行。 • Removed: 已移除。
target	否	String	**参数说明**：执行批量任务的目标，当task_type为firmwareUpgrade，softwareUpgrade，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows，此处填写device_id **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。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-561 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

响应参数

状态码： 200

表 1-562 响应 Body 参数

参数	参数类型	描述
batchtask	**Task** object	批量任务基本信息。
task_details	Array of **TaskDetail** objects	子任务详情列表。
page	**Page** object	查询结果的分页信息。

表 1-563 Task

参数	参数类型	描述
task_id	String	批量任务ID，创建批量任务时由物联网平台分配获得。
task_name	String	批量任务名称。

参数	参数类型	描述
task_type	String	批量任务类型，取值范围：firmwareUpgrade，softwareUpgrade，createDevices，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows。 ● softwareUpgrade: 软件升级任务 ● firmwareUpgrade: 固件升级任务 ● createDevices: 批量创建设备任务 ● deleteDevices: 批量删除设备任务 ● freezeDevices: 批量冻结设备任务 ● unfreezeDevices: 批量解冻设备任务 ● createCommands: 批量创建同步命令任务 ● createAsyncCommands: 批量创建异步命令任务 ● createMessages: 批量创建消息任务 ● updateDeviceShadows: 批量配置设备影子任务 ● updateDevices：批量更新设备任务
task_mode	String	**参数说明**：批量任务的模式，当前只支持网关模式，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，若升级的设备为某个网关的子设备，则平台下发获取版本信息通知和平台下发升级通知将携带task_id（软固件升级批量任务的任务ID）和sub_device_count（批量任务中网关设备包含的升级子设备数量）字段。 **取值范围**：GATEWAY: 网关模式。
task_ext_info	Object	**参数说明**：批量任务额外扩展信息，当task_type为firmwareUpgrade，softwareUpgrade支持该参数。软固件升级的场景下，平台下发获取版本信息通知和平台下发升级通知将携带该字段。 **取值范围**：最长不超过512个字符。 最大长度：**512**
targets	Array of strings	执行批量任务的目标，当task_type为firmwareUpgrade，softwareUpgrade，deleteDevices，freezeDevices，unfreezeDevices，createCommands，createAsyncCommands，createMessages，updateDeviceShadows，此处填写device_id列表。

参数	参数类型	描述
targets_filter	Map<String,Object>	任务目标筛选参数。Json格式，里面是一个个键值对，（K,V）格式标识筛选targets需要的参数，目前支持的K有group_ids（V填写group_id数组，eg:["e495cf17-ff79-4294-8f64-4d367919d665"]，任务则会筛选出来符合该群组条件的设备作为目标）
document	Object	执行任务数据文档，Json格式。(当task_type为softwareUpgrade|firmwareUpgrade，也就是软固件升级任务时，Json里面是(K,V)键值对，需要填写key为package_id，value为在平台上传的软固件附件id，id由portal软件库包管理上传并查询获得。当task_type为createCommands，也就是批量创建同步命令任务时，Json里面是命令相关参数，eg：{"service_id":"water","command_name":"ON_OFF","paras":{"value":"ON"}}，参考**设备同步命令**)。当task_type为createAsyncCommands，也就是批量创建异步命令任务时，Json里面是命令相关参数，eg：{"service_id":"water","command_name":"ON_OFF","paras":{"value":"ON"},"expire_time":0,"send_strategy":"immediately"}，参考**设备异步命令**)。当task_type为updateDeviceShadows，也就是批量配置设备影子任务时，Json里面是命令相关参数，eg：{"shadow": [{"service_id":"WaterMeter","desired": {"temperature":"60"}}]}，参考**配置设备影子预期数据**)。
task_policy	**TaskPolicy** object	任务执行策略。
status	String	批量任务的状态，可选参数，取值范围：Success|Fail|Processing|PartialSuccess|Stopped|Waitting|Initializing|Stopping。 • Initializing: 初始化中。 • Waitting: 等待中。 • Processing: 执行中。 • Success: 成功。 • Fail: 失败。 • PartialSuccess: 部分成功。 • Stopped: 停止。 • Stopping 停止中。
status_desc	String	批量任务状态描述(包含主任务失败错误信息)
task_progress	**TaskProgress** object	子任务执行统计结果。

参数	参数类型	描述
create_time	String	批量任务的创建时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-564 TaskPolicy

参数	参数类型	描述
schedule_time	String	**参数说明**：批量任务指定执行时间。 **取值范围**：7天内，不传入此参数表示立即执行，格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
retry_count	Integer	**参数说明**：批量任务子任务自动重试次数。 **取值范围**：如果传入retry_interval参数，则需传入该参数，最大支持重试5次。 最小值：**1** 最大值：**5**
retry_interval	Integer	**参数说明**：批量任务子任务失败后，自动重试时间间隔，单位：分钟。 **取值范围**：最大1440(24小时)，不传入此参数表示不重试，如果传入retry_count参数则需要传入该参数。 最小值：**0** 最大值：**1440**

表 1-565 TaskProgress

参数	参数类型	描述
total	Integer	子任务总个数。
processing	Integer	正在执行的子任务个数。
success	Integer	执行成功的子任务个数。
fail	Integer	执行失败的的子任务个数。
waitting	Integer	等待执行的子任务个数。
fail_wait_retry	Integer	失败等待重试的子任务个数。
stopped	Integer	停止的子任务个数。
removed	Integer	移除的子任务个数。

表 1-566 TaskDetail

参数	参数类型	描述
target	String	执行批量任务的目标。
status	String	子任务的执行状态，结果范围：Processing，Success，Fail，Waitting，FailWaitRetry，Stopped。 • Waitting: 等待执行。 • Processing: 执行中。 • Success: 成功。 • Fail: 失败。 • FailWaitRetry: 失败重试。 • Stopped: 已停止。
output	String	子任务执行的输出信息。
error	**ErrorInfo** object	子任务执行失败信息。

表 1-567 ErrorInfo

参数	参数类型	描述
error_code	String	错误码
error_msg	String	错误描述

表 1-568 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

查询指定批量任务详情。

```
GET https://{endpoint}/v5/iot/{project_id}/batchtasks/{taskId}
```

响应示例

状态码： 200

OK


```
{
  "batchtask" : {
    "task_id" : "5c8ba99030344005c02316ad",
    "task_name" : "testname",
    "task_type" : "softwareUpgrade",
    "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ],
    "targets_filter" : {
      "group_ids" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
    },
    "document" : {
      "package_id" : "32822e5744a45ede319d2c50"
    },
    "task_policy" : {
      "schedule_time" : "20151212T121212Z",
      "retry_count" : 5,
      "retry_interval" : 60
    },
    "status" : "Success",
    "status_desc" : "string",
    "task_progress" : {
      "total" : 0,
      "processing" : 0,
      "success" : 0,
      "fail" : 0,
      "waitting" : 0,
      "fail_wait_retry" : 0,
      "stopped" : 0
    },
    "create_time" : "20151212T121212Z"
  },
  "task_details" : [ {
    "target" : "e495cf17-ff79-4294-8f64-4d367919d665",
    "status" : "Success",
    "output" : "success",
    "error" : {
      "error_code" : "IOTDA.000002",
      "error_msg" : "The request is unauthorized."
    }
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```


SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
```

```
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowBatchTaskRequest request = new ShowBatchTaskRequest();
        request.withTaskDetailStatus("<task_detail_status>");
        request.withTarget("<target>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ShowBatchTaskResponse response = client.showBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowBatchTaskRequest()
        request.task_detail_status = "<task_detail_status>"
        request.target = "<target>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.show_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowBatchTaskRequest{}
    taskDetailStatusRequest:= "<task_detail_status>"
    request.TaskDetailStatus = &taskDetailStatusRequest
    targetRequest:= "<target>"
    request.Target = &targetRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
```



```
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ShowBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.4 删除批量任务

功能介绍

应用服务器可调用此接口删除物联网平台中已经完成（状态为成功，失败，部分成功，已停止）的批量任务。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/batchtasks/{task_id}

表 1-569 路径参数

参数	是否必选	参数类型	描述
task_id	是	String	**参数说明**：批量任务ID，创建批量任务时由物联网平台分配获得。 **取值范围**：长度不超过24，只允许小写字母a到f、数字的组合。
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-570 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

响应参数

无

请求示例

删除批量任务。

```
DELETE https://{endpoint}/v5/iot/{project_id}/batchtasks/{taskId}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteBatchTaskRequest request = new DeleteBatchTaskRequest();
        try {
            DeleteBatchTaskResponse response = client.deleteBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```



```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteBatchTaskRequest()
        response = client.delete_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteBatchTaskRequest{}
    response, err := client.DeleteBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.5 重试批量任务

功能介绍

应用服务器可调用此接口重试批量任务，目前只支持task_type为firmwareUpgrade，softwareUpgrade。如果task_id对应任务已经成功、停止、正在停止、等待中或初始化中，则不可以调用该接口。如果请求Body为{}，则调用该接口后会重新执行所有状态为失败、失败待重试和已停止的子任务。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/batchtasks/{task_id}/retry

表 1-571 路径参数

参数	是否必选	参数类型	描述
task_id	是	String	**参数说明**：批量任务ID，创建批量任务时由物联网平台分配获得。 **取值范围**：长度不超过24，只允许小写字母a到f、数字的组合。
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-572 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

表 1-573 请求 Body 参数

参数	是否必选	参数类型	描述
targets	否	Array of strings	执行批量任务的目标集合，最多支持100个目标，当task_type为firmwareUpgrade，softwareUpgrade时，此处填写device_id

响应参数

状态码： 200

表 1-574 响应 Body 参数

参数	参数类型	描述
targets	Array of **BatchTargetResult** objects	批量操作目标结果集合

表 1-575 BatchTargetResult

参数	参数类型	描述
target	String	执行批量任务的目标。
status	String	目标的执行结果，为success或failure
error_code	String	操作失败的错误码
error_msg	String	操作失败的错误描述

请求示例

- 重新执行状态为失败，失败待重试和已停止的子任务

```
POST https://{endpoint}/v5/iot/{project_id}/batchtasks/{task_id}/retry

{ }
```

- 重新执行指定目标集合的子任务

```
POST https://{endpoint}/v5/iot/{project_id}/batchtasks/{task_id}/retry

{
  "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
}
```

响应示例

状态码： 200

OK

```
{
  "targets" : [ {
    "target" : "e495cf17-ff79-4294-8f64-4d367919d665",
    "status" : "failure",
    "error_code" : "IOTDA.014219",
    "error_msg" : "Invalid input. The target is not in the task"
  }, {
    "target" : "e495cf17-ff79-4294-8f64-4d367919d677",
    "status" : "success"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 重新执行状态为失败，失败待重试和已停止的子任务

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class RetryBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        RetryBatchTaskRequest request = new RetryBatchTaskRequest();
        BatchTargets body = new BatchTargets();
        request.withBody(body);
        try {
            RetryBatchTaskResponse response = client.retryBatchTask(request);
            System.out.println(response.toString());
```

```
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 重新执行指定目标集合的子任务

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class RetryBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        RetryBatchTaskRequest request = new RetryBatchTaskRequest();
        BatchTargets body = new BatchTargets();
        List<String> listbodyTargets = new ArrayList<>();
        listbodyTargets.add("e495cf17-ff79-4294-8f64-4d367919d665");
        body.withTargets(listbodyTargets);
        request.withBody(body);
        try {
            RetryBatchTaskResponse response = client.retryBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
```

```
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 重新执行状态为失败，失败待重试和已停止的子任务

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = RetryBatchTaskRequest()
        request.body = BatchTargets(
        )
        response = client.retry_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 重新执行指定目标集合的子任务

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
```

```
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = RetryBatchTaskRequest()
        listTargetsbody = [
            "e495cf17-ff79-4294-8f64-4d367919d665"
        ]
        request.body = BatchTargets(
            targets=listTargetsbody
        )
        response = client.retry_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 重新执行状态为失败，失败待重试和已停止的子任务

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.RetryBatchTaskRequest{}
    request.Body = &model.BatchTargets{
    }
    response, err := client.RetryBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 重新执行指定目标集合的子任务

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.RetryBatchTaskRequest{}
    var listTargetsbody = []string{
        "e495cf17-ff79-4294-8f64-4d367919d665",
    }
    request.Body = &model.BatchTargets{
        Targets: &listTargetsbody,
    }
    response, err := client.RetryBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```



```
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.6 停止批量任务

功能介绍

应用服务器可调用此接口停止批量任务，目前只支持task_type为firmwareUpgrade，softwareUpgrade。如果task_id对应任务已经完成（成功、失败、部分成功，已经停止）或正在停止中，则不可以调用该接口。如果请求Body为{}，则调用该接口后会停止所有正在执行中、等待中和失败待重试状态的子任务。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/batchtasks/{task_id}/stop

表 1-576 路径参数

参数	是否必选	参数类型	描述
task_id	是	String	**参数说明**：批量任务ID，创建批量任务时由物联网平台分配获得。 **取值范围**：长度不超过24，只允许小写字母a到f、数字的组合。
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-577 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

表 1-578 请求 Body 参数

参数	是否必选	参数类型	描述
targets	否	Array of strings	执行批量任务的目标集合，最多支持100个目标，当task_type为firmwareUpgrade，softwareUpgrade时，此处填写device_id

响应参数

状态码： 200

表 1-579 响应 Body 参数

参数	参数类型	描述
targets	Array of **BatchTargetResult** objects	批量操作目标结果集合

表 1-580 BatchTargetResult

参数	参数类型	描述
target	String	执行批量任务的目标。
status	String	目标的执行结果，为success或failure
error_code	String	操作失败的错误码
error_msg	String	操作失败的错误描述

请求示例

- 停止所有正在执行中，等待中和失败待重试状态的子任务。

```
POST https://{endpoint}/v5/iot/{project_id}/batchtasks/{task_id}/stop

{ }
```

- 停止执行指定目标集合的子任务

```
POST https://{endpoint}/v5/iot/{project_id}/batchtasks/{task_id}/stop

{
  "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
}
```

响应示例

状态码： 200

OK

```
{
  "targets" : [ {
    "target" : "e495cf17-ff79-4294-8f64-4d367919d665",
    "status" : "failure",
    "error_code" : "IOTDA.014219",
    "error_msg" : "Invalid input. The target is not in the task"
  }, {
    "target" : "e495cf17-ff79-4294-8f64-4d367919d677",
    "status" : "success"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 停止所有正在执行中，等待中和失败待重试状态的子任务。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class StopBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        StopBatchTaskRequest request = new StopBatchTaskRequest();
        BatchTargets body = new BatchTargets();
        request.withBody(body);
        try {
            StopBatchTaskResponse response = client.stopBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 停止执行指定目标集合的子任务


```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class StopBatchTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        StopBatchTaskRequest request = new StopBatchTaskRequest();
        BatchTargets body = new BatchTargets();
        List<String> listbodyTargets = new ArrayList<>();
        listbodyTargets.add("e495cf17-ff79-4294-8f64-4d367919d665");
        body.withTargets(listbodyTargets);
        request.withBody(body);
        try {
            StopBatchTaskResponse response = client.stopBatchTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 停止所有正在执行中，等待中和失败待重试状态的子任务。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = StopBatchTaskRequest()
        request.body = BatchTargets(
        )
        response = client.stop_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 停止执行指定目标集合的子任务

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
```

```
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = StopBatchTaskRequest()
        listTargetsbody = [
            "e495cf17-ff79-4294-8f64-4d367919d665"
        ]
        request.body = BatchTargets(
            targets=listTargetsbody
        )
        response = client.stop_batch_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 停止所有正在执行中，等待中和失败待重试状态的子任务。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.StopBatchTaskRequest{}
    request.Body = &model.BatchTargets{
    }
    response, err := client.StopBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
```



```
        fmt.Println(err)
    }
}
```

- 停止执行指定目标集合的子任务

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.StopBatchTaskRequest{}
    var listTargetsbody = []string{
        "e495cf17-ff79-4294-8f64-4d367919d665",
    }
    request.Body = &model.BatchTargets{
        Targets: &listTargetsbody,
    }
    response, err := client.StopBatchTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.7 批量任务的文件管理

1.4.15.7.1 上传批量任务文件

功能介绍

应用服务器可调用此接口上传批量任务文件，用于创建批量任务。当前支持批量创建设备任务、批量删除设备任务、批量冻结设备任务、批量解冻设备任务的文件上传。

- **批量注册设备模板**
- **批量删除设备模板**
- **批量冻结设备模板**
- **批量解冻设备模板**

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/batchtask-files

表 1-581 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-582 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

表 1-583 FormData 参数

参数	是否必选	参数类型	描述
file	是	File	**参数说明**：上传批量任务文件。 **取值范围**：当前仅支持xlsx/xls文件格式，且文件最大行数为100000行。

响应参数

状态码： 201

表 1-584 响应 Body 参数

参数	参数类型	描述
file_id	String	上传的批量任务文件ID，由平台自动生成。
file_name	String	上传的批量任务文件名称。 最小长度：**1** 最大长度：**60**

参数	参数类型	描述
upload_time	String	在物联网平台上传文件的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

请求示例

上传批量任务文件。

```
POST https://{endpoint}/v5/iot/{project_id}/batchtask-files
```

响应示例

状态码： 201

Created

```
{
  "file_id" : "0c3c77dd-42a2-4309-9e10-da2e8bf64ac3",
  "file_name" : "BatchCreateDevices_test01.xlsx",
  "upload_time" : "20200617T081608Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UploadBatchTaskFileSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
```



```
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UploadBatchTaskFileRequest request = new UploadBatchTaskFileRequest();
        try {
            UploadBatchTaskFileResponse response = client.uploadBatchTaskFile(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UploadBatchTaskFileRequest()
        response = client.upload_batch_task_file(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UploadBatchTaskFileRequest{}
    response, err := client.UploadBatchTaskFile(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	BAD REQUEST
401	Unauthorized
403	FORBIDDEN

状态码	描述
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.7.2 查询批量任务文件列表

功能介绍

应用服务器可调用此接口查询批量任务文件列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/batchtask-files

表 1-585 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-586 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。

参数	是否必选	参数类型	描述
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

响应参数

状态码： 200

表 1-587 响应 Body 参数

参数	参数类型	描述
files	Array of **BatchTaskFile** objects	批量任务文件列表。 数组长度：**0 - 10**

表 1-588 BatchTaskFile

参数	参数类型	描述
file_id	String	上传的批量任务文件ID，由平台自动生成。
file_name	String	上传的批量任务文件名称。 最小长度：**1** 最大长度：**60**
upload_time	String	在物联网平台上传文件的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

请求示例

列表查询批量任务文件。

```
GET https://{endpoint}/v5/iot/{project_id}/batchtask-files
```

响应示例

状态码： 200

OK

```
{
  "files" : [ {
    "file_id" : "0c3c77dd-42a2-4309-9e10-da2e8bf64ac3",
    "file_name" : "BatchCreateDevices_test01.xlsx",
    "upload_time" : "20200617T081608Z"
  }, {
    "file_id" : "9c338eba-b162-4005-98ea-ff34a13c70da",
    "file_name" : "BatchCreateDevices_test02.xlsx",
    "upload_time" : "20200617T081620Z"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListBatchTaskFilesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListBatchTaskFilesRequest request = new ListBatchTaskFilesRequest();
        try {
            ListBatchTaskFilesResponse response = client.listBatchTaskFiles(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
```

```
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListBatchTaskFilesRequest()
        response = client.list_batch_task_files(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
```

```
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListBatchTaskFilesRequest{}
    response, err := client.ListBatchTaskFiles(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.15.7.3 删除批量任务文件

功能介绍

应用服务器可调用此接口删除批量任务文件。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/batchtask-files/{file_id}

表 1-589 路径参数

参数	是否必选	参数类型	描述
file_id	是	String	**参数说明**：要删除的批量任务文件ID。 **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。 最小长度：**1** 最大长度：**128**
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-590 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。 最小长度：**1** 最大长度：**36**

响应参数

无

请求示例

删除批量任务文件。

```
DELETE https://{endpoint}/v5/iot/{project_id}/batchtask-files/{file_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteBatchTaskFileSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteBatchTaskFileRequest request = new DeleteBatchTaskFileRequest();
        try {
            DeleteBatchTaskFileResponse response = client.deleteBatchTaskFile(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
```

```
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteBatchTaskFileRequest()
        response = client.delete_batch_task_file(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
```

```
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteBatchTaskFileRequest{}
    response, err := client.DeleteBatchTaskFile(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.16 设备 CA 证书管理

1.4.16.1 上传设备 CA 证书

功能介绍

应用服务器可调用此接口在物联网平台上传设备CA证书

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/certificates

表 1-591 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-592 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。。
Instance-Id	否	String	实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-593 请求 Body 参数

参数	是否必选	参数类型	描述
content	是	String	证书内容信息。 最小长度：**1** 最大长度：**65535**
app_id	否	String	资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的证书归属到哪个资源空间下，否则创建的证书将会归属到**默认资源空间**下。

响应参数

状态码： 201

表 1-594 响应 Body 参数

参数	参数类型	描述
certificate_id	String	CA证书ID，在上传CA证书时由平台分配的唯一标识。
cn_name	String	CA证书CN名称。
owner	String	CA证书所有者。
status	Boolean	CA证书验证状态。true代表证书已通过验证，可进行设备证书认证接入。false代表证书未通过验证。
verify_code	String	CA证书验证码。
provision_enable	Boolean	是否开启自注册能力，当为true时该功能必须配合预调配功能使用，true：是，false：否。
template_id	String	绑定的预调配模板ID。
create_date	String	创建证书日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
effective_date	String	CA证书生效日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
expiry_date	String	CA证书失效日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

请求示例

上传设备CA证书。

```
POST https://{endpoint}/v5/iot/{project_id}/certificates

{
  "content" : "-----BEGINCERTIFICATE-----
\nMIID2TCCAsGgAwIBAgIJAOEDEgVdVMn9MA0GCSqGSIb3DQEBCwUAMIGCMQswCQYD
\nVQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcxETAPBgNVBAcMCFNoZW56aGVuMQ8w
\nDQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lvdDESMBAGA1UEAwwJMTIzNDU2Nzg5\nMRowGAYJKoZ
IhvcNAQkBFgtkamthQHFxLmNvbTAeFw0xOTEyMTkxMzE1MjZaFw0y
\nMjEwMDgxMzE1MjZaMIGCMQswCQYDVQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcx
\nETAPBgNVBAcMCFNoZW56aGVuMQ8wDQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lv
\ndDESMBAGA1UEAwwJMTIzNDU2Nzg5MRowGAYJKoZIhvcNAQkBFgtkamthQHFxLmNv
\nbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAM72QUzoadvLfxGjt3UF
\noZ4MJbblqnRbouO4KpOVHBXyS2yQVl4CWWMhLh4pp2efNUSqKuXHjY3r68PquyNn
\nYk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJIpRzkmlCBbxwTNCjc3FovGb/sHHNlpGn
\ncCKUzMfPGNZuBiuemskuEXL/eMHxDPbXYWn4Wq0wt+28PKUL5jybY7nsXSNnmAPF
\nTO0CAmq0meUukubT/jHDCQ78ihQ/iqw1RNq88aCqRleoHiGg5nWkjL+05GXqUrqV\nVnZNL
+YqcXzuVMs5XgyhNM2AsuH2g3D8ZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnF\nD/
0CAwEAAaNQME4wHQYDVR0OBBYEFAVPWVtpTdO6KQnmVrrNlMguWNR7MB8GA1Ud
\nIwQYMBaAFAVPWVtpTdO6KQnmVrrNlMguWNR7MAwGA1UdEwQFMAMBAf8wDQYJKoZI
\nhvcNAQELBQADggEBAE40ViqK+UaEn++Xq6f4Cmeg3JqYHu47v9RIAASNihYRBQ/r
\n3RE7Af3GqjIO5nMJJuCMzdcoAU8N9KwkgXD+GLR9fYLEoEmq5CrhgaGDsCi85vCs
\nmWhj5z8r5TG207xpmvH2KT447dnG+chMBE594ma85dCv+0mCDrqNToElipgT8+rY
\nAYVClnIt3kbsTg1vSRNHadd+TpgRVxJZBF0fHcCAyc/2f3UJgPYNWShIetHM6BdI\n3fZ4H
+eeHPjagm5kzmffli1cUv2/N+1hKUvcl4uFCqEwZRFtp90RyIbxUfQwi+Cs\nXVnwV
+BZS5qD9bTcfxZMXhuVRwO/5xWYMYPN1uY=\n-----END CERTIFICATE-----",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
}
```

响应示例

状态码： 201

Created

```
{
  "certificate_id" : "string",
  "cn_name" : "string",
  "owner" : "string",
  "status" : true,
  "verify_code" : "string",
  "create_date" : "20191212T121212Z",
  "effective_date" : "20191212T121212Z",
  "expiry_date" : "20221212T121212Z",
  "provision_enable" : true,
  "template_id" : "61c970ce2d63eb6ee655dbf0"
}
```

SDK 代码示例

SDK代码示例如下。

Java

上传设备CA证书。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
```



```
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class AddCertificateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddCertificateRequest request = new AddCertificateRequest();
        CreateCertificateDTO body = new CreateCertificateDTO();
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withContent("-----BEGINCERTIFICATE-----
MIID2TCCAsGgAwIBAgIJAOEDEgVdVMn9MA0GCSqGSIb3DQEBCwUAMIGCMQswCQYD
VQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcxETAPBgNVBAcMCFNoZW56aGVuMQ8w
DQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lvdDESMBAGA1UEAwwJMTIzNDU2Nzg5
MRowGAYJKoZIhvcNAQkBFgtkamthQHFxLmNvbTAeFw0xOTEyMTkxMzE1MjZaFw0y
MjEwMDgxMzE1MjZaMIGCMQswCQYDVQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcx
ETAPBgNVBAcMCFNoZW56aGVuMQ8wDQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lv
dDESMBAGA1UEAwwJMTIzNDU2Nzg5MRowGAYJKoZIhvcNAQkBFgtkamthQHFxLmNv
bTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAM72QUzoadvLfxGjt3UF
oZ4MJbblqnRbouO4KpOVHBXyS2yQVl4CWWMhLh4pp2efNUSqKuXHjY3r68PquyNn
Yk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJIpRzkmlCBbxwTNCjc3FovGb/sHHNlpGn
cCKUzMfPGNZuBiuemskuEXL/eMHxDPbXYWn4Wq0wt+28PKUL5jybY7nsXSNnmAPF
TO0CAmq0meUukubT/jHDCQ78ihQ/iqw1RNq88aCqRleoHiGg5nWkjL+05GXqUrqV
VnZNL+YqcXzuVMs5XgyhNM2AsuH2g3D8ZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnF
D/0CAwEAAaNQME4wHQYDVR0OBBYEFAVPWVtpTdO6KQnmVrrNlMguWNR7MB8GA1Ud
IwQYMBaAFAVPWVtpTdO6KQnmVrrNlMguWNR7MAwGA1UdEwQFMAMBAf8wDQYJKoZI
hvcNAQELBQADggEBAE40ViqK+UaEn++Xq6f4Cmeg3JqYHu47v9RIAASNihYRBQ/r
3RE7Af3GqjIO5nMJJuCMzdcoAU8N9KwkgXD+GLR9fYLEoEmq5CrhgaGDsCi85vCs
mWhj5z8r5TG207xpmvH2KT447dnG+chMBE594ma85dCv+0mCDrqNToElipgT8+rY
AYVClnIt3kbsTg1vSRNHadd+TpgRVxJZBF0fHcCAyc/2f3UJgPYNWShIetHM6BdI
3fZ4H+eeHPjagm5kzmffli1cUv2/N+1hKUvcI4uFCqEwZRFtp90RyIbxUfQwi+Cs
XVnwV+BZS5qD9bTcfxZMXhuVRwO/5xWYMYPN1uY=
-----END CERTIFICATE-----");
        request.withBody(body);
        try {
            AddCertificateResponse response = client.addCertificate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
```

```
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

上传设备CA证书。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddCertificateRequest()
        request.body = CreateCertificateDTO(
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            content="-----BEGINCERTIFICATE-----
MIID2TCCAsGgAwIBAgIJAOEDEgVdVMn9MA0GCSqGSIb3DQEBCwUAMIGCMQswCQYD
VQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcxETAPBgNVBAcMCFNoZW56aGVuMQ8w
DQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lvdDESMBAGA1UEAwwJMTIzNDU2Nzg5
MRowGAYJKoZIhvcNAQkBFgtkamthQHFxLmNvbTAeFw0xOTEyMTkxMzE1MjZaFw0y
MjEwMDgxMzE1MjZaMIGCMQswCQYDVQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcx
ETAPBgNVBAcMCFNoZW56aGVuMQ8wDQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lv
dDESMBAGA1UEAwwJMTIzNDU2Nzg5MRowGAYJKoZIhvcNAQkBFgtkamthQHFxLmNv
bTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAM72QUzoadvLfxGjt3UF
oZ4MJbblqnRbouO4KpOVHBXyS2yQVl4CWWMhLh4pp2efNUSqKuXHjY3r68PquyNn
Yk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJIpRzkmlCBbxwTNCjc3FovGb/sHHNlpGn
cCKUzMfPGNZuBiuemskuEXL/eMHxDPbXYWn4Wq0wt+28PKUL5jybY7nsXSNnmAPF
TO0CAmq0meUukubT/jHDCQ78ihQ/iqw1RNq88aCqRleoHiGg5nWkjL+05GXqUrqV
VnZNL+YqcXzuVMs5XgyhNM2AsuH2g3D8ZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnF
D/0CAwEAAaNQME4wHQYDVR0OBBYEFAVPWVtpTdO6KQnmVrrNlMguWNR7MB8GA1Ud
IwQYMBaAFAVPWVtpTdO6KQnmVrrNlMguWNR7MAwGA1UdEwQFMAMBAf8wDQYJKoZI
hvcNAQELBQADggEBAE40ViqK+UaEn++Xq6f4Cmeg3JqYHu47v9RIAASNihYRBQ/r
3RE7Af3GqjIO5nMJJuCMzdcoAU8N9KwkgXD+GLR9fYLEoEmq5CrhgaGDsCi85vCs
mWhj5z8r5TG207xpmvH2KT447dnG+chMBE594ma85dCv+0mCDrqNToElipgT8+rY
AYVClnIt3kbsTg1vSRNHadd+TpgRVxJZBF0fHcCAyc/2f3UJgPYNWShIetHM6BdI
3fZ4H+eeHPjagm5kzmffli1cUv2/N+1hKUvcI4uFCqEwZRFtp90RyIbxUfQwi+Cs
XVnwV+BZS5qD9bTcfxZMXhuVRwO/5xWYMYPN1uY=
-----END CERTIFICATE-----"
        )
```



```
        response = client.add_certificate(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

上传设备CA证书。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddCertificateRequest{}
    appIdCreateCertificateDto:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    request.Body = &model.CreateCertificateDto{
        AppId: &appIdCreateCertificateDto,
        Content: "-----BEGINCERTIFICATE-----
MIID2TCCAsGgAwIBAgIJAOEDEgVdVMn9MA0GCSqGSIb3DQEBCwUAMIGCMQswCQYD
VQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcxETAPBgNVBAcMCFNoZW56aGVuMQ8w
DQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lvdDESMBAGA1UEAwwJMTIzNDU2Nzg5
MRowGAYJKoZIhvcNAQkBFgtkamthQHFxLmNvbTAeFw0xOTEyMTkxMzE1MjZaFw0y
MjEwMDgxMzE1MjZaMIGCMQswCQYDVQQGEwJDTjERMA8GA1UECAwIR3VhbmRvbmcx
ETAPBgNVBAcMCFNoZW56aGVuMQ8wDQYDVQQKDAZIdWF3ZWkxDDAKBgNVBAsMA2lv
dDESMBAGA1UEAwwJMTIzNDU2Nzg5MRowGAYJKoZIhvcNAQkBFgtkamthQHFxLmNv
bTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAM72QUzoadvLfxGjt3UF
oZ4MJbblqnRbouO4KpOVHBXyS2yQVl4CWWMhLh4pp2efNUSqKuXHjY3r68PquyNn
Yk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJIpRzkmlCBbxwTNCjc3FovGb/sHHNlpGn
cCKUzMfPGNZuBiuemskuEXL/eMHxDPbXYWn4Wq0wt+28PKUL5jybY7nsXSNnmAPF
TO0CAmq0meUukubT/jHDCQ78ihQ/iqw1RNq88aCqRleoHiGg5nWkjL+05GXqUrqV
VnZNL+YqcXzuVMs5XgyhNM2AsuH2g3D8ZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnF
D/0CAwEAAaNQME4wHQYDVR0OBBYEFAVPWVtpTdO6KQnmVrrNlMguWNR7MB8GA1Ud
IwQYMBaAFAVPWVtpTdO6KQnmVrrNlMguWNR7MAwGA1UdEwQFMAMBAf8wDQYJKoZI
```

```
hvcNAQELBQADggEBAE40ViqK+UaEn++Xq6f4Cmeg3JqYHu47v9RIAASNihYRBQ/r
3RE7Af3GqjIO5nMJJuCMzdcoAU8N9KwkgXD+GLR9fYLEoEmq5CrhgaGDsCi85vCs
mWhj5z8r5TG207xpmvH2KT447dnG+chMBE594ma85dCv+0mCDrqNToElipgT8+rY
AYVClnIt3kbsTg1vSRNHadd+TpgRVxJZBF0fHcCAyc/2f3UJgPYNWShIetHM6BdI
3fZ4H+eeHPjagm5kzmffli1cUv2/N+1hKUvcI4uFCqEwZRFtp90RyIbxUfQwi+Cs
XVnwV+BZS5qD9bTcfxZMXhuVRwO/5xWYMYPN1uY=
-----END CERTIFICATE-----",
    }
    response, err := client.AddCertificate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
401	Unauthorized
403	Forbidden
400	Bad Request
500	Internal Server Error

错误码

请参见**错误码**。

1.4.16.2 获取设备 CA 证书列表

功能介绍

应用服务器可调用此接口在物联网平台获取设备CA证书列表

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/certificates

表 1-595 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID。获取方法请参见 **获取项目ID** 。

表 1-596 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的证书列表，不携带该参数则会查询该用户下所有证书列表。
limit	否	Integer	分页查询时每页显示的记录数，默认值为10，取值范围为1-50的整数。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 缺省值： **ffffffffffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-597 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。。
Instance-Id	否	String	实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-598 响应 Body 参数

参数	参数类型	描述
certificates	Array of **CertificatesRspDTO** objects	证书列表。
page	**Page** object	查询结果的分页信息。

表 1-599 CertificatesRspDTO

参数	参数类型	描述
certificate_id	String	CA证书ID，在上传CA证书时由平台分配的唯一标识。
cn_name	String	CA证书CN名称。
owner	String	CA证书所有者。
status	Boolean	CA证书验证状态。true代表证书已通过验证，可进行设备证书认证接入。false代表证书未通过验证。
verify_code	String	CA证书验证码。
provision_enable	Boolean	是否开启自注册能力，当为true时该功能必须配合预调配功能使用，true：是，false：否。
template_id	String	绑定的预调配模板ID。
create_date	String	创建证书日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
effective_date	String	CA证书生效日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
expiry_date	String	CA证书失效日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-600 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

获取设备CA证书列表。

```
GET https://{endpoint}/v5/iot/{project_id}/certificates
```

响应示例

状态码： 200

OK

```
{
  "certificates" : [ {
    "certificate_id" : "string",
    "cn_name" : "string",
    "owner" : "string",
    "status" : true,
    "verify_code" : "string",
    "create_date" : "20191212T121212Z",
    "effective_date" : "20191212T121212Z",
    "expiry_date" : "20221212T121212Z",
    "provision_enable" : true,
    "template_id" : "61c970ce2d63eb6ee655dbf0"
  } ],
  "page" : {
    "count" : 100,
    "marker" : "5c8f3d2d3df1f10d803adbda"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListCertificatesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
```

```
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListCertificatesRequest request = new ListCertificatesRequest();
        request.withAppId("<app_id>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListCertificatesResponse response = client.listCertificates(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListCertificatesRequest()
        request.app_id = "<app_id>"
        request.limit = <limit>
        request.marker = "<marker>"
```

```
        request.offset = <offset>
        response = client.list_certificates(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListCertificatesRequest{}
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListCertificates(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.16.3 删除设备 CA 证书

功能介绍

应用服务器可调用此接口在物联网平台删除设备CA证书

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/certificates/{certificate_id}

表 1-601 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID。获取方法请参见 **获取项目ID** 。
certificate_id	是	String	设备CA证书ID，在上传设备CA证书时由平台分配的唯一标识。 最小长度：**1** 最大长度：**36**

请求参数

表 1-602 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。。
Instance-Id	否	String	实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除设备CA证书。

```
DELETE https://{endpoint}/v5/iot/{project_id}/certificates/{certificate_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteCertificateSolution {
```



```
    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteCertificateRequest request = new DeleteCertificateRequest();
        try {
            DeleteCertificateResponse response = client.deleteCertificate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
```



```
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteCertificateRequest()
        response = client.delete_certificate(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteCertificateRequest{}
    response, err := client.DeleteCertificate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.16.4 更新 CA 证书

功能介绍

应用服务器可调用此接口在物联网平台上更新CA证书。仅标准版实例、企业版实例支持该接口调用，基础版不支持。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/certificates/{certificate_id}

表 1-603 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID。获取方法请参见 **获取项目ID** 。
certificate_id	是	String	CA证书ID，在上传CA证书时由平台分配的唯一标识。 最小长度：**1** 最大长度：**36**

请求参数

表 1-604 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。。
Instance-Id	否	String	实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-605 请求 Body 参数

参数	是否必选	参数类型	描述
provision_enable	否	Boolean	是否开启自注册能力，当为true时该功能必须配合预调配功能使用，true：是，false：否。
template_id	否	String	预调配模板ID，该CA证书绑定的预调配模板id，当该字段传null时表示解除绑定关系。

响应参数

状态码： 200

表 1-606 响应 Body 参数

参数	参数类型	描述
certificate_id	String	CA证书ID，在上传CA证书时由平台分配的唯一标识。
cn_name	String	CA证书CN名称。
owner	String	CA证书所有者。
status	Boolean	CA证书验证状态。true代表证书已通过验证，可进行设备证书认证接入。false代表证书未通过验证。

参数	参数类型	描述
verify_code	String	CA证书验证码。
provision_enable	Boolean	是否开启自注册能力，当为true时该功能必须配合预调配功能使用，true：是，false：否。
template_id	String	绑定的预调配模板ID。
create_date	String	创建证书日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
effective_date	String	CA证书生效日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
expiry_date	String	CA证书失效日期。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

请求示例

证书关联自注册模板，并开启自注册功能。

```
PUT https://{endpoint}/v5/iot/{project_id}/certificates/{certificate_id}

{
  "template_id" : "61c970ce2d63eb6ee655dbf0",
  "provision_enable" : true
}
```

响应示例

状态码： 200

Successful response

```
{
  "certificate_id" : "string",
  "cn_name" : "string",
  "owner" : "string",
  "status" : true,
  "verify_code" : "string",
  "provision_enable" : true,
  "template_id" : "61c970ce2d63eb6ee655dbf0",
  "create_date" : "20191212T121212Z",
  "effective_date" : "20191212T121212Z",
  "expiry_date" : "20221212T121212Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

证书关联自注册模板，并开启自注册功能。


```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateCertificateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateCertificateRequest request = new UpdateCertificateRequest();
        UpdateCertificateDTO body = new UpdateCertificateDTO();
        body.withTemplateId("61c970ce2d63eb6ee655dbf0");
        body.withProvisionEnable(true);
        request.withBody(body);
        try {
            UpdateCertificateResponse response = client.updateCertificate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

证书关联自注册模板，并开启自注册功能。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
```

```
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateCertificateRequest()
        request.body = UpdateCertificateDTO(
            template_id="61c970ce2d63eb6ee655dbf0",
            provision_enable=True
        )
        response = client.update_certificate(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

证书关联自注册模板，并开启自注册功能。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```



```
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateCertificateRequest{}
    templateIdUpdateCertificateDto:= "61c970ce2d63eb6ee655dbf0"
    provisionEnableUpdateCertificateDto:= true
    request.Body = &model.UpdateCertificateDto{
        TemplateId: &templateIdUpdateCertificateDto,
        ProvisionEnable: &provisionEnableUpdateCertificateDto,
    }
    response, err := client.UpdateCertificate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Successful response
400	Bad Request
401	Unauthorized
404	Not Found
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.16.5 验证设备 CA 证书

功能介绍

应用服务器可调用此接口在物联网平台验证设备的CA证书，目的是为了验证用户持有设备CA证书的私钥

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/certificates/{certificate_id}/action

表 1-607 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID。获取方法请参见 **获取项目ID** 。
certificate_id	是	String	设备CA证书ID，在上传设备CA证书时由平台分配的唯一标识。 最小长度：**1** 最大长度：**36**

表 1-608 Query 参数

参数	是否必选	参数类型	描述
action_id	是	String	对证书执行的操作，当前仅支持verify:校验证书

请求参数

表 1-609 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。。
Instance-Id	否	String	实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-610 请求 Body 参数

参数	是否必选	参数类型	描述
verify_content	是	String	验证证书的内容信息。 最小长度：**1** 最大长度：**65535**

响应参数

无

请求示例

验证设备CA证书。

```
POST https://{endpoint}/v5/iot/{project_id}/certificates/{certificate_id}/action

{
  "verify_content" : "-----BEGIN CERTIFICATE-----
\nMIIDnzCCAocCCQCs5+qyyItl5TANBgkqhkiG9w0BAQsFADCBgjELMAkGA1UEBhMC
\nQ04xETAPBgNVBAgMCEd1YW5kb25nMREwDwYDVQQHDAhTaGVuemhlbjEPMA0GA1UE
\nCgwGSHVhd2VpMQwwCgYDVQQLDANpb3QxEjAQBgNVBAMMCTEyMzQ1Njc4OTEaMBgG
\nCSqGSIb3DQEJARYLZGprYUBxcS5jb20wHhcNMTkxMjE5MTMyMTM3WhcNMjEwNTAy
\nMTMyMTM3WjCBnzELMAkGA1UEBhMCQ04xEjAQBgNVBAgMCUd1YW5nZG9uZzERMA8G
\nA1UEBwwIU2hlbnpoZW4xDzANBgNVBAoMBm9yaWdpbjENMAsGA1UECwwEdW5pdDEt
\nMCsGA1UEAwwkMmM4YjU5MDUtYjM0YS00YjY0LTgxMTItZjZjMDQ3YWUwNjVjMRow
\nGAYJKoZIhvcNAQkBFgtqbGtqQHFxLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP
\nADCCAQoCggEBAM72QUzoadvLfxGjt3UFoZ4MJbblqnRbouO4KpOVHBXyS2yQVl4C
\nWWMhLh4pp2efNUSqKuXHjY3r68PquyNnYk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJ
\nIpRzkmlCBbxwTNCjc3FovGb/sHHNlpGncCKUzMfPGNZuBiuemskuEXL/eMHxDPbX\nYWn4Wq0wt
+28PKUL5jybY7nsXSNnmAPFTO0CAmq0meUukubT/jHDCQ78ihQ/iqw1\nRNq88aCqRleoHiGg5nWkjL
+05GXqUrqVVnZNL+YqcXzuVMs5XgyhNM2AsuH2g3D8\nZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnFD/
0CAwEAATANBgkqhkiG9w0BAQsFAAOC\nAQEAh1SF1Z/
p8nT7k8868lLNBZrIcErMlkFdghn2HRYyw5iilDXL28lJEBax2X1M\nNl2fD/rov9gwxhyrBZD2YkevL8k
+DXcVpVEoozwpUR3p79YEyT0E3jI67G/EiB2h\n+o7+deDlH7d7Li/ZOSQC6JTSLshBhi
+B8CQmYYt6YCjN7Rswbf1Z8bsQNrcsxW36\nZM3uG3i9GrEktypTNXMRUbG5gngaFKbRGGUPWNYdNXQeXU
W9cpj8HAyndESEwAYz\ntLKHdnM874P8ZAmRkijZoToOCMcT0s8l8SoYUR7iWI0E08KYzAPgLX9Xvw42GCEF
\nb2TJfnOIwhu8gFf7cwlCGC+gRA==\n-----END CERTIFICATE-----"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

验证设备CA证书。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
```



```
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CheckCertificateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CheckCertificateRequest request = new CheckCertificateRequest();
        request.withActionId("<action_id>");
        VerifyCertificateDTO body = new VerifyCertificateDTO();
        body.withVerifyContent("-----BEGIN CERTIFICATE-----
MIIDnzCCAocCCQCs5+qyyItl5TANBgkqhkiG9w0BAQsFADCBgjELMAkGA1UEBhMC
Q04xETAPBgNVBAgMCEd1YW5kb25nMREwDwYDVQQHDAhTaGVuemhlbjEPMA0GA1UE
CgwGSHVhd2VpMQwwCgYDVQQLDANpb3QxEjAQBgNVBAMMCTEyMzQ1Njc4OTEaMBgG
CSqGSIb3DQEJARYLZGprYUBxcS5jb20wHhcNMTkxMjE5MTMyMTM3WhcNMjEwNTAy
MTMyMTM3WjCBnzELMAkGA1UEBhMCQ04xEjAQBgNVBAgMCUd1YW5nZG9uZzERMA8G
A1UEBwwIU2hlbnpoZW4xDzANBgNVBAoMBm9yaWdpbjENMAsGA1UECwwEdW5pdDEt
MCsGA1UEAwwkMmM4YjU5MDUtYjM0YS00YjY0LTgxMTItZjZjMDQ3YWUwNjVjMRow
GAYJKoZIhvcNAQkBFgtqbGtqQHFxLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP
ADCCAQoCggEBAM72QUzoadvLfxGjt3UFoZ4MJbblqnRbouO4KpOVHBXyS2yQVl4C
WWMhLh4pp2efNUSqKuXHjY3r68PquyNnYk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJ
IpRzkmlCBbxwTNCjc3FovGb/sHHNlpGncCKUzMfPGNZuBiuemskuEXL/eMHxDPbX
YWn4Wq0wt+28PKUL5jybY7nsXSNnmAPFTO0CAmq0meUukubT/jHDCQ78ihQ/iqw1
RNq88aCqRleoHiGg5nWkjL+05GXqUrqVVnZNL+YqcXzuVMs5XgyhNM2AsuH2g3D8
ZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnFD/0CAwEAATANBgkqhkiG9w0BAQsFAAOC
AQEAh1SF1Z/p8nT7k8868lLNBZrIcErMlkFdghn2HRYyw5iilDXL28lJEBax2X1M
Nl2fD/rov9gwxhyrBZD2YkevL8k+DXcVpVEoozwpUR3p79YEyT0E3jI67G/EiB2h
+o7+deDlH7d7Li/ZOSQC6JTSLshBhi+B8CQmYYt6YCjN7Rswbf1Z8bsQNrcsxW36
ZM3uG3i9GrEktypTNXMRUbG5gngaFKbRGGUPWNYdNXQeXUW9cpj8HAyndESEwAYz
tLKHdnM874P8ZAmRkijZoToOCMcT0s8l8SoYUR7iWI0E08KYzAPgLX9Xvw42GCEF
b2TJfnOIwhu8gFf7cwlCGC+gRA==
-----END CERTIFICATE-----");
        request.withBody(body);
        try {
            CheckCertificateResponse response = client.checkCertificate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
```

```
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

验证设备CA证书。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CheckCertificateRequest()
        request.action_id = "<action_id>"
        request.body = VerifyCertificateDTO(
            verify_content="-----BEGIN CERTIFICATE-----
MIIDnzCCAocCCQCs5+qyyItl5TANBgkqhkiG9w0BAQsFADCBgjELMAkGA1UEBhMC
Q04xETAPBgNVBAgMCEd1YW5kb25nMREwDwYDVQQHDAhTaGVuemhlbjEPMA0GA1UE
CgwGSHVhd2VpMQwwCgYDVQQLDANpb3QxEjAQBgNVBAMMCTEyMzQ1Njc4OTEaMBgG
CSqGSIb3DQEJARYLZGprYUBxcS5jb20wHhcNMTkxMjE5MTMyMTM3WhcNMjEwNTAy
MTMyMTM3WjCBnzELMAkGA1UEBhMCQ04xEjAQBgNVBAgMCUd1YW5nZG9uZzERMA8G
A1UEBwwIU2hlbnpoZW4xDzANBgNVBAoMBm9yaWdpbjENMAsGA1UECwwEdW5pdDEt
MCsGA1UEAwwkMmM4YjU5MDUtYjM0YS00YjY0LTgxMTItZjZjMDQ3YWUwNjVjMRow
GAYJKoZIhvcNAQkBFgtqbGtqQHFxLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP
ADCCAQoCggEBAM72QUzoadvLfxGjt3UFoZ4MJbblqnRbouO4KpOVHBXyS2yQVl4C
WWMhLh4pp2efNUSqKuXHjY3r68PquyNnYk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJ
IpRzkmlCBbxwTNCjc3FovGb/sHHNlpGncCKUzMfPGNZuBiuemskuEXL/eMHxDPbX
YWn4Wq0wt+28PKUL5jybY7nsXSNnmAPFTO0CAmq0meUukubT/jHDCQ78ihQ/iqw1
RNq88aCqRleoHiGg5nWkjL+05GXqUrqVVnZNL+YqcXzuVMs5XgyhNM2AsuH2g3D8
ZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnFD/0CAwEAATANBgkqhkiG9w0BAQsFAAOC
AQEAh1SF1Z/p8nT7k8868lLNBZrIcErMlkFdghn2HRYyw5iilDXL28lJEBax2X1M
Nl2fD/rov9gwxhyrBZD2YkevL8k+DXcVpVEoozwpUR3p79YEyT0E3jI67G/EiB2h
+o7+deDlH7d7Li/ZOSQC6JTSLshBhi+B8CQmYYt6YCjN7Rswbf1Z8bsQNrcsxW36
ZM3uG3i9GrEktypTNXMRUbG5gngaFKbRGGUPWNYdNXQeXUW9cpj8HAyndESEwAYz
tLKHdnM874P8ZAmRkijZoToOCMcT0s8l8SoYUR7iWI0E08KYzAPgLX9Xvw42GCEF
b2TJfnOIwhu8gFf7cwlCGC+gRA==
-----END CERTIFICATE-----"
        )
        response = client.check_certificate(request)
```



```
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

验证设备CA证书。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CheckCertificateRequest{}
    request.ActionId = "<action_id>"
    request.Body = &model.VerifyCertificateDto{
        VerifyContent: "-----BEGIN CERTIFICATE-----
MIIDnzCCAocCCQCs5+qyyItl5TANBgkqhkiG9w0BAQsFADCBgjELMAkGA1UEBhMC
Q04xETAPBgNVBAgMCEd1YW5kb25nMREwDwYDVQQHDAhTaGVuemhlbjEPMA0GA1UE
CgwGSHVhd2VpMQwwCgYDVQQLDANpb3QxEjAQBgNVBAMMCTEyMzQ1Njc4OTEaMBgG
CSqGSIb3DQEJARYLZGprYUBxcS5jb20wHhcNMTkxMjE5MTMyMTM3WhcNMjEwNTAy
MTMyMTM3WjCBnzELMAkGA1UEBhMCQ04xEjAQBgNVBAgMCUd1YW5nZG9uZzERMA8G
A1UEBwwIU2hlbnpoZW4xDzANBgNVBAoMBm9yaWdpbjENMAsGA1UECwwEdW5pdDEt
MCsGA1UEAwwkMmM4YjU5MDUtYjM0YS00YjY0LTgxMTItZjZjMDQ3YWUwNjVjMRow
GAYJKoZIhvcNAQkBFgtqbGtqQHFxLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP
ADCCAQoCggEBAM72QUzoadvLfxGjt3UFoZ4MJbblqnRbouO4KpOVHBXyS2yQVl4C
WWMhLh4pp2efNUSqKuXHjY3r68PquyNnYk8zO59zVc7JHvjGkBvo7DgPRAhEKPLJ
IpRzkmlCBbxwTNCjc3FovGb/sHHNlpGncCKUzMfPGNZuBiuemskuEXL/eMHxDPbX
YWn4Wq0wt+28PKUL5jybY7nsXSNnmAPFTO0CAmq0meUukubT/jHDCQ78ihQ/iqw1
RNq88aCqRleoHiGg5nWkjL+05GXqUrqVVnZNL+YqcXzuVMs5XgyhNM2AsuH2g3D8
ZuF6Dj9qY1n/v/Cp/DGpxP3A74SlplnFD/0CAwEAATANBgkqhkiG9w0BAQsFAAOC
AQEAh1SF1Z/p8nT7k8868lLNBZrIcErMlkFdghn2HRYyw5iilDXL28lJEBax2X1M
Nl2fD/rov9gwxhyrBZD2YkevL8k+DXcVpVEoozwpUR3p79YEyT0E3jI67G/EiB2h
+o7+deDlH7d7Li/ZOSQC6JTSLshBhi+B8CQmYYt6YCjN7Rswbf1Z8bsQNrcsxW36
```

```
ZM3uG3i9GrEktypTNXMRUbG5gngaFKbRGGUPWNYdNXQeXUW9cpj8HAyndESEwAYz
tLKHdnM874P8ZAmRkijZoToOCMcT0s8l8SoYUR7iWI0E08KYzAPgLX9Xvw42GCEF
b2TJfnOIwhu8gFf7cwlCGC+gRA==
-----END CERTIFICATE-----",
    }
    response, err := client.CheckCertificate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.17 OTA 升级包管理

1.4.17.1 创建 OTA 升级包

功能介绍

用户可调用此接口创建升级包关联OBS对象 使用前提：使用该API需要您授权设备接入服务(IoTDA)的实例访问对象存储服务(OBS)以及 密钥管理服务(KMS Administrator)的权限。在“**统一身份认证服务（IAM）** - 委托”中将委托名称为iotda_admin_trust的委托授权KMS Administrator和OBS OperateAccess

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/ota-upgrades/packages

表 1-611 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-612 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-613 请求 Body 参数

参数	是否必选	参数类型	描述
app_id	是	String	**参数说明**：资源空间ID。存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的升级包归属到哪个资源空间下。**取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
package_type	是	String	**参数说明**：升级包类型。 **取值范围**：软件包必须设置为：softwarePackage，固件包必须设置为：firmwarePackage。

参数	是否必选	参数类型	描述
product_id	是	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
version	是	String	**参数说明**：升级包版本号。 **取值范围**：长度不超过256，只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
support_source_versions	否	Array of strings	**参数说明**：支持用于升级此版本包的设备源版本号列表。最多支持20个源版本号。 **取值范围**：源版本号列表，源版本号只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
description	否	String	**参数说明**：用于描述升级包的功能等信息。 **取值范围**：长度不超过1024。 最大长度：**1024**
custom_info	否	String	**参数说明**：推送给设备的自定义信息。添加该升级包完成，并创建升级任务后，物联网平台向设备下发升级通知时，会下发该自定义信息给设备。 **取值范围**：长度不超过4096。 最大长度：**4096**
file_location	是	**FileLocation** object	升级包的位置

表 1-614 FileLocation

参数	是否必选	参数类型	描述
obs_location	否	**ObsLocation** object	升级包关联OBS对象位置

表 1-615 ObsLocation

参数	是否必选	参数类型	描述
region_name	是	String	**参数说明**：OBS所在区域。您可以从**地区和终端节点**中查询服务的终端节点。 **取值范围**：长度不超过256，只允许字母、数字、连接符（-）的组合。
bucket_name	是	String	**参数说明**：OBS桶名称。 **取值范围**：长度最小为3，最大为63，只允许小写字母、数字、连接符（-）、英文点（.）的组合。
object_key	是	String	**参数说明**：OBS对象名称(包含文件夹路径)，对象大小最大为1G，且只支持.bin、.dav、.tar、.gz、.zip、.gzip、.apk、.tar.gz、.tar.xz、.pack、.exe、.bat、.img格式的文件。 **取值范围**：长度不超过1024。 最小长度：**1** 最大长度：**1024**
sign	否	String	**参数说明**：SHA256算法计算出的升级包签名值。添加该升级包完成，并创建升级任务后，物联网平台向设备下发升级通知时，会下发该签名给设备。 **取值范围**：长度为64，只允许大小写字母a到f、数字的组合。

响应参数

状态码： 201

表 1-616 响应 Body 参数

参数	参数类型	描述
package_id	String	**参数说明**：升级包ID，用于唯一标识一个升级包。由物联网平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、连接符（-）的组合。
app_id	String	**参数说明**：资源空间ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

参数	参数类型	描述
package_type	String	**参数说明**：升级包类型。 **取值范围**：软件包必须设置为：softwarePackage，固件包必须设置为：firmwarePackage。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
version	String	**参数说明**：升级包版本号。 **取值范围**：长度不超过256，只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
support_source_versions	Array of strings	**参数说明**：支持用于升级此版本包的设备源版本号列表。最多支持20个源版本号。 **取值范围**：源版本号列表，源版本号只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
description	String	**参数说明**：用于描述升级包的功能等信息。 **取值范围**：长度不超过1024。
custom_info	String	**参数说明**：推送给设备的自定义信息。添加该升级包完成，并创建升级任务后，物联网平台向设备下发升级通知时，会下发该自定义信息给设备。 **取值范围**：长度不超过4096。
create_time	String	软固件包上传到物联网平台的时间，格式："yyyyMMdd'T'HHmmss'Z'"。
file_location	**FileLocation** object	升级包的位置

表 1-617 FileLocation

参数	参数类型	描述
obs_location	**ObsLocation** object	升级包关联OBS对象位置

表 1-618 ObsLocation

参数	参数类型	描述
region_name	String	**参数说明**：OBS所在区域。您可以从**地区和终端节点**中查询服务的终端节点。 **取值范围**：长度不超过256，只允许字母、数字、连接符（-）的组合。

参数	参数类型	描述
bucket_name	String	**参数说明**：OBS桶名称。 **取值范围**：长度最小为3，最大为63，只允许小写字母、数字、连接符（-）、英文点（.）的组合。
object_key	String	**参数说明**：OBS对象名称(包含文件夹路径)，对象大小最大为1G，且只支持.bin、.dav、.tar、.gz、.zip、.gzip、.apk、.tar.gz、.tar.xz、.pack、.exe、.bat、.img格式的文件。 **取值范围**：长度不超过1024。 最小长度：**1** 最大长度：**1024**
sign	String	**参数说明**：SHA256算法计算出的升级包签名值。添加该升级包完成，并创建升级任务后，物联网平台向设备下发升级通知时，会下发该签名给设备。 **取值范围**：长度为64，只允许大小写字母a到f、数字的组合。

请求示例

- 创建OTA升级包，上传固件包。

```
POST https://{endpoint}/v5/iot/{project_id}/ota-upgrades/packages

{
  "app_id" : "61f7e74d036aca5be29e1ed4",
  "package_type" : "firmwarePackage",
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "version" : "V2.0",
  "description" : "package v2.0",
  "custom_info" : "更新了XX功能，修复了XXXX问题",
  "file_location" : {
    "obs_location" : {
      "region_name" : "cn-north-4",
      "bucket_name" : "abc",
      "object_key" : "bbb/upgrade.bin"
    }
  }
}
```

- 创建OTA升级包，上传固件包，差分包场景，支持从V1.0和V1.1版本升级。

```
POST https://{endpoint}/v5/iot/{project_id}/ota-upgrades/packages

{
  "app_id" : "61f7e74d036aca5be29e1ed4",
  "package_type" : "firmwarePackage",
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "version" : "V2.0",
  "support_source_versions" : [ "V1.0", "V1.1" ],
  "description" : "package for version V1.0 and V1.1",
  "custom_info" : "更新了XX功能，修复了XXXX问题",
  "file_location" : {
    "obs_location" : {
      "region_name" : "cn-north-4",
      "bucket_name" : "abc",
      "object_key" : "bbb/upgrade.bin"
    }
  }
}
```


响应示例

状态码： 201

Created

```
{
  "package_id" : "28f61af50fc9452aa0ed5ea25c3cc3d3",
  "app_id" : "61f7e74d036aca5be29e1ed4",
  "package_type" : "firmwarePackage",
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "version" : "V2.0",
  "support_source_versions" : [ "V1.0", "V1.1" ],
  "description" : "package for version V1.0 and V1.1",
  "custom_info" : "更新了XX功能，修复了XXXX问题",
  "create_time" : "20230211T121212Z",
  "file_location" : {
    "obs_location" : {
      "region_name" : "cn-north-4",
      "bucket_name" : "abc",
      "object_key" : "bbb/upgrade.bin"
    }
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建OTA升级包，上传固件包。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateOtaPackageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
```



```
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateOtaPackageRequest request = new CreateOtaPackageRequest();
        CreateOtaPackage body = new CreateOtaPackage();
        ObsLocation obsLocationFileLocation = new ObsLocation();
        obsLocationFileLocation.withRegionName("cn-north-4")
            .withBucketName("abc")
            .withObjectKey("bbb/upgrade.bin");
        FileLocation fileLocationbody = new FileLocation();
        fileLocationbody.withObsLocation(obsLocationFileLocation);
        body.withFileLocation(fileLocationbody);
        body.withCustomInfo("更新了XX功能,修复了XXXX问题");
        body.withDescription("package v2.0");
        body.withVersion("V2.0");
        body.withProductId("5ba24f5ebbe8f56f5a14f605");
        body.withPackageType("firmwarePackage");
        body.withAppId("61f7e74d036aca5be29e1ed4");
        request.withBody(body);
        try {
            CreateOtaPackageResponse response = client.createOtaPackage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建OTA升级包，上传固件包，差分包场景，支持从V1.0和V1.1版本升级。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateOtaPackageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
```

```
            .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
            .withAk(ak)
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        CreateOtaPackageRequest request = new CreateOtaPackageRequest();
        CreateOtaPackage body = new CreateOtaPackage();
        ObsLocation obsLocationFileLocation = new ObsLocation();
        obsLocationFileLocation.withRegionName("cn-north-4")
            .withBucketName("abc")
            .withObjectKey("bbb/upgrade.bin");
        FileLocation fileLocationbody = new FileLocation();
        fileLocationbody.withObsLocation(obsLocationFileLocation);
        List<String> listbodySupportSourceVersions = new ArrayList<>();
        listbodySupportSourceVersions.add("V1.0");
        listbodySupportSourceVersions.add("V1.1");
        body.withFileLocation(fileLocationbody);
        body.withCustomInfo("更新了XX功能,修复了XXXX问题");
        body.withDescription("package for version V1.0 and V1.1");
        body.withSupportSourceVersions(listbodySupportSourceVersions);
        body.withVersion("V2.0");
        body.withProductId("5ba24f5ebbe8f56f5a14f605");
        body.withPackageType("firmwarePackage");
        body.withAppId("61f7e74d036aca5be29e1ed4");
        request.withBody(body);
        try {
            CreateOtaPackageResponse response = client.createOtaPackage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建OTA升级包，上传固件包。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
```

```
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateOtaPackageRequest()
        obsLocationFileLocation = ObsLocation(
            region_name="cn-north-4",
            bucket_name="abc",
            object_key="bbb/upgrade.bin"
        )
        fileLocationbody = FileLocation(
            obs_location=obsLocationFileLocation
        )
        request.body = CreateOtaPackage(
            file_location=fileLocationbody,
            custom_info="更新了XX功能,修复了XXXX问题",
            description="package v2.0",
            version="V2.0",
            product_id="5ba24f5ebbe8f56f5a14f605",
            package_type="firmwarePackage",
            app_id="61f7e74d036aca5be29e1ed4"
        )
        response = client.create_ota_package(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建OTA升级包，上传固件包，差分包场景，支持从V1.0和V1.1版本升级。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
```

```
        .build()

    try:
        request = CreateOtaPackageRequest()
        obsLocationFileLocation = ObsLocation(
            region_name="cn-north-4",
            bucket_name="abc",
            object_key="bbb/upgrade.bin"
        )
        fileLocationbody = FileLocation(
            obs_location=obsLocationFileLocation
        )
        listSupportSourceVersionsbody = [
            "V1.0",
            "V1.1"
        ]
        request.body = CreateOtaPackage(
            file_location=fileLocationbody,
            custom_info="更新了XX功能,修复了XXXX问题",
            description="package for version V1.0 and V1.1",
            support_source_versions=listSupportSourceVersionsbody,
            version="V2.0",
            product_id="5ba24f5ebbe8f56f5a14f605",
            package_type="firmwarePackage",
            app_id="61f7e74d036aca5be29e1ed4"
        )
        response = client.create_ota_package(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建OTA升级包，上传固件包。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
```

```
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateOtaPackageRequest{}
    obsLocationFileLocation := &model.ObsLocation{
        RegionName: "cn-north-4",
        BucketName: "abc",
        ObjectKey: "bbb/upgrade.bin",
    }
    fileLocationbody := &model.FileLocation{
        ObsLocation: obsLocationFileLocation,
    }
    customInfoCreateOtaPackage:= "更新了XX功能,修复了XXXX问题"
    descriptionCreateOtaPackage:= "package v2.0"
    request.Body = &model.CreateOtaPackage{
        FileLocation: fileLocationbody,
        CustomInfo: &customInfoCreateOtaPackage,
        Description: &descriptionCreateOtaPackage,
        Version: "V2.0",
        ProductId: "5ba24f5ebbe8f56f5a14f605",
        PackageType: "firmwarePackage",
        AppId: "61f7e74d036aca5be29e1ed4",
    }
    response, err := client.CreateOtaPackage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建OTA升级包，上传固件包，差分包场景，支持从V1.0和V1.1版本升级。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
```

```
        WithCredential(auth).
        Build())

    request := &model.CreateOtaPackageRequest{}
    obsLocationFileLocation := &model.ObsLocation{
        RegionName: "cn-north-4",
        BucketName: "abc",
        ObjectKey: "bbb/upgrade.bin",
    }
    fileLocationbody := &model.FileLocation{
        ObsLocation: obsLocationFileLocation,
    }
    var listSupportSourceVersionsbody = []string{
        "V1.0",
        "V1.1",
    }
    customInfoCreateOtaPackage:= "更新了XX功能,修复了XXXX问题"
    descriptionCreateOtaPackage:= "package for version V1.0 and V1.1"
    request.Body = &model.CreateOtaPackage{
        FileLocation: fileLocationbody,
        CustomInfo: &customInfoCreateOtaPackage,
        Description: &descriptionCreateOtaPackage,
        SupportSourceVersions: &listSupportSourceVersionsbody,
        Version: "V2.0",
        ProductId: "5ba24f5ebbe8f56f5a14f605",
        PackageType: "firmwarePackage",
        AppId: "61f7e74d036aca5be29e1ed4",
    }
    response, err := client.CreateOtaPackage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.17.2 查询 OTA 升级包列表

功能介绍

用户可调用此接口查询关联OBS对象的升级包列表 使用前提：使用该API需要您授权设备接入服务(IoTDA)的实例访问对象存储服务(OBS)以及 密钥管理服务(KMS Administrator)的权限。在“**统一身份认证服务（IAM）** - 委托”中将委托名称为iotda_admin_trust的委托授权KMS Administrator和OBS OperateAccess

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/ota-upgrades/packages

表 1-619 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-620 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。存在多资源空间的用户需要使用该接口时，建议携带该参数指定查询指定资源空间的升级包列表。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
package_type	是	String	**参数说明**：升级包类型。 **取值范围**：软件包必须设置为：softwarePackage，固件包必须设置为：firmwarePackage。
product_id	否	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

参数	是否必选	参数类型	描述
version	否	String	**参数说明**：升级包版本号。 **取值范围**：长度不超过256，只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。**取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-621 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-622 响应 Body 参数

参数	参数类型	描述
packages	Array of **OtaPackageInfo** objects	升级包列表
page	**PageInfo** object	查询批量分页结构体，定义了分页页码、每页记录数、记录总数、该页记录的最大Id。

表 1-623 OtaPackageInfo

参数	参数类型	描述
package_id	String	**参数说明**：升级包ID，用于唯一标识一个升级包。由物联网平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、连接符（-）的组合。
app_id	String	**参数说明**：资源空间ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
package_type	String	**参数说明**：升级包类型。 **取值范围**：软件包必须设置为：softwarePackage，固件包必须设置为：firmwarePackage。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
version	String	**参数说明**：升级包版本号。 **取值范围**：长度不超过256，只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
support_source_versions	Array of strings	**参数说明**：支持用于升级此版本包的设备源版本号列表。最多支持20个源版本号。 **取值范围**：源版本号列表，源版本号只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
description	String	**参数说明**：用于描述升级包的功能等信息。 **取值范围**：长度不超过1024。
custom_info	String	**参数说明**：推送给设备的自定义信息。添加该升级包完成，并创建升级任务后，物联网平台向设备下发升级通知时，会下发该自定义信息给设备。 **取值范围**：长度不超过4096。
create_time	String	软固件包上传到物联网平台的时间，格式："yyyyMMdd'T'HHmmss'Z'"。

表 1-624 PageInfo

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

列表查询OTA升级包。

```
GET https://{endpoint}/v5/iot/{project_id}/ota-upgrades/packages
```

响应示例

状态码： 200

OK

```
{
  "packages" : [ {
    "package_id" : "28f61af50fc9452aa0ed5ea25c3cc3d3",
    "app_id" : "61f7e74d036aca5be29e1ed4",
    "package_type" : "firmwarePackage",
    "product_id" : "5ba24f5ebbe8f56f5a14f605",
    "version" : "V2.0",
    "support_source_versions" : [ "V1.0", "V1.1" ],
    "description" : "package for version V1.0 and V1.1",
    "custom_info" : "更新了XX功能，修复了XXXX问题",
    "create_time" : "20230211T121212Z"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListOtaPackageInfoSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
```



```
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListOtaPackageInfoRequest request = new ListOtaPackageInfoRequest();
        request.withAppId("<app_id>");
        request.withPackageType("<package_type>");
        request.withProductId("<product_id>");
        request.withVersion("<version>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListOtaPackageInfoResponse response = client.listOtaPackageInfo(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";
```



```
    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListOtaPackageInfoRequest()
        request.app_id = "<app_id>"
        request.package_type = "<package_type>"
        request.product_id = "<product_id>"
        request.version = "<version>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_ota_package_info(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListOtaPackageInfoRequest{}
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
```



```
    request.PackageType = "<package_type>"
    productIdRequest:= "<product_id>"
    request.ProductId = &productIdRequest
    versionRequest:= "<version>"
    request.Version = &versionRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListOtaPackageInfo(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	BAD REQUEST
401	Unauthorized
403	FORBIDDEN
500	Internal Server Error

错误码

请参见**错误码**。

1.4.17.3 获取 OTA 升级包详情

功能介绍

用户可调用此接口查询关联OBS对象的升级包详情 使用前提：使用该API需要您授权设备接入服务(IoTDA)的实例访问对象存储服务(OBS)以及 密钥管理服务(KMS Administrator)的权限。在“**统一身份认证服务（IAM）** - 委托”中将委托名称为iotda_admin_trust的委托授权KMS Administrator和OBS OperateAccess

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/ota-upgrades/packages/{package_id}

表 1-625 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
package_id	是	String	**参数说明**：升级包ID，用于唯一标识一个升级包。由物联网平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、连接符（-）的组合。

请求参数

表 1-626 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-627 响应 Body 参数

参数	参数类型	描述
package_id	String	**参数说明**：升级包ID，用于唯一标识一个升级包。由物联网平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、连接符（-）的组合。
app_id	String	**参数说明**：资源空间ID。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
package_type	String	**参数说明**：升级包类型。 **取值范围**：软件包必须设置为：softwarePackage，固件包必须设置为：firmwarePackage。
product_id	String	**参数说明**：设备关联的产品ID，用于唯一标识一个产品模型，创建产品后获得。方法请参见 **创建产品**。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
version	String	**参数说明**：升级包版本号。 **取值范围**：长度不超过256，只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
support_source_versions	Array of strings	**参数说明**：支持用于升级此版本包的设备源版本号列表。最多支持20个源版本号。 **取值范围**：源版本号列表，源版本号只允许字母、数字、下划线（_）、连接符（-）、英文点（.）的组合。
description	String	**参数说明**：用于描述升级包的功能等信息。 **取值范围**：长度不超过1024。
custom_info	String	**参数说明**：推送给设备的自定义信息。添加该升级包完成，并创建升级任务后，物联网平台向设备下发升级通知时，会下发该自定义信息给设备。 **取值范围**：长度不超过4096。
create_time	String	软固件包上传到物联网平台的时间，格式："yyyyMMdd'T'HHmmss'Z'"。
file_location	**FileLocation** object	升级包的位置

表 1-628 FileLocation

参数	参数类型	描述
obs_location	**ObsLocation** object	升级包关联OBS对象位置

表 1-629 ObsLocation

参数	参数类型	描述
region_name	String	**参数说明**：OBS所在区域。您可以从**地区和终端节点**中查询服务的终端节点。 **取值范围**：长度不超过256，只允许字母、数字、连接符（-）的组合。
bucket_name	String	**参数说明**：OBS桶名称。 **取值范围**：长度最小为3，最大为63，只允许小写字母、数字、连接符（-）、英文点（.）的组合。
object_key	String	**参数说明**：OBS对象名称(包含文件夹路径)，对象大小最大为1G，且只支持.bin、.dav、.tar、.gz、.zip、.gzip、.apk、.tar.gz、.tar.xz、.pack、.exe、.bat、.img格式的文件。 **取值范围**：长度不超过1024。 最小长度：**1** 最大长度：**1024**
sign	String	**参数说明**：SHA256算法计算出的升级包签名值。添加该升级包完成，并创建升级任务后，物联网平台向设备下发升级通知时，会下发该签名给设备。 **取值范围**：长度为64，只允许大小写字母a到f、数字的组合。

请求示例

查询OTA升级包详情。

```
GET https://{endpoint}/v5/iot/{project_id}/ota-upgrades/packages/{package_id}
```

响应示例

状态码： 200

OK

```
{
  "package_id" : "28f61af50fc9452aa0ed5ea25c3cc3d3",
  "app_id" : "61f7e74d036aca5be29e1ed4",
  "package_type" : "firmwarePackage",
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "version" : "V2.0",
  "support_source_versions" : [ "V1.0", "V1.1" ],
  "description" : "package for version V1.0 and V1.1",
  "custom_info" : "更新了XX功能，修复了XXXX问题",
  "create_time" : "20230211T121212Z",
  "file_location" : {
    "obs_location" : {
      "region_name" : "cn-north-4",
      "bucket_name" : "abc",
      "object_key" : "bbb/upgrade.bin"
    }
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowOtaPackageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowOtaPackageRequest request = new ShowOtaPackageRequest();
        try {
            ShowOtaPackageResponse response = client.showOtaPackage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```



```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowOtaPackageRequest()
        response = client.show_ota_package(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```

```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowOtaPackageRequest{}
    response, err := client.ShowOtaPackage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	FORBIDDEN
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.17.4 删除 OTA 升级包

功能介绍

用户可调用此接口删除关联OBS对象的升级包信息，不会删除OBS上对象 使用前提：使用该API需要您授权设备接入服务(IoTDA)的实例访问对象存储服务(OBS)以及 密钥管理服务(KMS Administrator)的权限。在“**统一身份认证服务（IAM）** - 委托”中将委托名称为iotda_admin_trust的委托授权KMS Administrator和OBS OperateAccess

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/ota-upgrades/packages/{package_id}

表 1-630 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
package_id	是	String	**参数说明**：升级包ID，用于唯一标识一个升级包。由物联网平台分配获得。 **取值范围**：长度不超过36，只允许字母、数字、连接符（-）的组合。

请求参数

表 1-631 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除OTA升级包。

```
DELETE https://{endpoint}/v5/iot/{project_id}/ota-upgrades/packages/{package_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteOtaPackageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteOtaPackageRequest request = new DeleteOtaPackageRequest();
        try {
            DeleteOtaPackageResponse response = client.deleteOtaPackage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```


Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteOtaPackageRequest()
        response = client.delete_ota_package(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```

```
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteOtaPackageRequest{}
    response, err := client.DeleteOtaPackage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.18 广播消息

1.4.18.1 下发广播消息

功能介绍

应用服务器可调用此接口向订阅了指定Topic的所有在线设备发布广播消息。应用将广播消息下发给平台后，平台会先返回应用响应结果，再将消息广播给设备。 注意：

- 此接口只适用于使用MQTT协议接入的设备。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/broadcast-messages

表 1-632 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-633 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-634 请求 Body 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定广播消息所属的资源空间，否则广播消息将会向**默认资源空间**下订阅指定topic的设备发送。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
topic_full_name	是	String	**参数说明**：接收广播消息的完整Topic名称, 必选。用户需要发布广播消息给设备时，可以使用该参数指定完整的topic名称，物联网平台会向指定资源空间下订阅了该topic的所有在线设备发送消息。广播的topic无需控制台创建，但topic的前缀必须为$oc/broadcast/ 最大长度：**128**
message	是	String	**参数说明**：广播消息的内容，用户需要将消息原文使用Base64编码。请求参数体最大长度为256KB。 最大长度：**261848**
ttl	否	Integer	**参数说明**：广播消息在平台缓存的老化时间，时间单位是分钟，默认值为0, 即默认不缓存消息；ttl>0时表示缓存消息，ttl参数数值必须是5的倍数，即以5分钟为粒度，默认最大缓存时间为1440分钟，指定的缓存时间超过1440分钟需申请，否则下发失败；ttl>0时，一个topic订阅设备数限制为10，如果一个topic订阅设备数超过10，则接口返回错误。 最小值：**0** 最大值：**10080** 缺省值：**0**

参数	是否必选	参数类型	描述
message_id	否	String	**参数说明**：消息id，由用户生成（推荐使用UUID）。ttl> 0时，平台会缓存消息，需确保message_id是唯一的， 否则接口返回错误。 **取值范围**：最大长度为128，只允许字母、数字、下划线（_）、连接符（-）的组合。 最大长度：**128**

响应参数

状态码： 201

表 1-635 响应 Body 参数

参数	参数类型	描述
app_id	String	**参数说明**：资源空间ID。
topic_full_name	String	**参数说明**：接收广播消息的完整Topic名称 最大长度：**128**
message_id	String	消息id，由物联网平台生成，用于标识该消息。
created_time	String	消息的创建时间，"yyyyMMdd'T'HHmmss'Z'"格式的UTC字符串。

请求示例

下发广播消息。

```
POST https://{endpoint}/v5/iot/{project_id}/broadcast-messages

{
  "topic_full_name" : "$oc/broadcast/test",
  "message" : "SGVsbG9Xb3JsZA=="
}
```

响应示例

状态码： 201

Created

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "topic_full_name" : "$oc/broadcast/test",
  "message_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "created_time" : "20151212T121212Z"
}
```


SDK 代码示例

SDK代码示例如下。

Java

下发广播消息。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class BroadcastMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        BroadcastMessageRequest request = new BroadcastMessageRequest();
        DeviceBroadcastRequest body = new DeviceBroadcastRequest();
        body.withMessage("SGVsbG9Xb3JsZA==");
        body.withTopicFullName("$oc/broadcast/test");
        request.withBody(body);
        try {
            BroadcastMessageResponse response = client.broadcastMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```


Python

下发广播消息。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = BroadcastMessageRequest()
        request.body = DeviceBroadcastRequest(
            message="SGVsbG9Xb3JsZA==",
            topic_full_name="$oc/broadcast/test"
        )
        response = client.broadcast_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

下发广播消息。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
```

```
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.BroadcastMessageRequest{}
    request.Body = &model.DeviceBroadcastRequest{
        Message: "SGVsbG9Xb3JsZA==",
        TopicFullName: "$oc/broadcast/test",
    }
    response, err := client.BroadcastMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.19 设备隧道管理

1.4.19.1 创建设备隧道

功能介绍

用户可以通过该接口创建隧道（WebSocket协议），应用服务器和设备可以通过该隧道进行数据传输。

- 该API接口在基础版不支持。
- 该API调用后平台会向对应的MQTT/MQTTS设备下发隧道地址及密钥，同时给应用服务器也返回隧道地址及密钥，设备可以通过该地址及密钥创建WebSocket协议连接。
- 一个设备无法创建多个隧道。
- 具体应用可见“设备远程登录”功能，请参见**设备远程登录**。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/tunnels

表 1-636 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-637 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。

参数	是否必选	参数类型	描述
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-638 请求 Body 参数

参数	是否必选	参数类型	描述
device_id	是	String	**参数说明**：设备ID

响应参数

状态码： 201

表 1-639 响应 Body 参数

参数	参数类型	描述
tunnel_id	String	隧道ID
tunnel_access_token	String	鉴权信息
expires_in	Integer	鉴权信息的过期时间, 单位：秒 最小值：**0** 最大值：**86400000**
tunnel_uri	String	websocket接入地址 最小长度：**1** 最大长度：**2048**

请求示例

创建设备隧道，设备id为b64b7a625b84c1334befb648b_test。

```
POST https://{endpoint}/v5/iot/{project_id}/tunnels

{
  "device_id" : "b64b7a625b84c1334befb648b_test"
}
```


响应示例

状态码： 201

创建设备隧道成功

```
{
  "tunnel_id" : "d144a524-1997-4b99-94bf-f27128da8a34",
  "tunnel_access_token" : "MIIDkgYJKoZIhvcNAQcCoIIDgzCCXXXXX",
  "expires_in" : 86400,
  "tunnel_uri" : "wss://tunnel.st1.iotda-app.cn-XXX.myhuaweicloud.com/v5/iot/tunnels/XXX/source-connect"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建设备隧道，设备id为b64b7a625b84c1334befb648b_test。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class AddTunnelSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddTunnelRequest request = new AddTunnelRequest();
        AddTunnelDto body = new AddTunnelDto();
        body.withDeviceId("b64b7a625b84c1334befb648b_test");
        request.withBody(body);
        try {
            AddTunnelResponse response = client.addTunnel(request);
            System.out.println(response.toString());
```

```
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建设备隧道，设备id为b64b7a625b84c1334befb648b_test。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddTunnelRequest()
        request.body = AddTunnelDto(
            device_id="b64b7a625b84c1334befb648b_test"
        )
        response = client.add_tunnel(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建设备隧道，设备id为b64b7a625b84c1334befb648b_test。

```
package main

import (
```



```
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddTunnelRequest{}
    request.Body = &model.AddTunnelDto{
        DeviceId: "b64b7a625b84c1334befb648b_test",
    }
    response, err := client.AddTunnel(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	创建设备隧道成功
400	输入参数不正确
401	Unauthorized
403	鉴权认证失败
500	Internal Server Error

错误码

请参见**错误码**。

1.4.19.2 查询设备所有隧道

功能介绍

用户可通过该接口查询某项目下的所有设备隧道，以实现对设备管理。应用服务器可通过此接口向平台查询设备隧道建立的情况。

- 该API接口在基础版不支持。
- 具体应用可见“设备远程登录”功能，请参见**设备远程登录**。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/tunnels

表 1-640 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-641 Query 参数

参数	是否必选	参数类型	描述
device_id	否	String	**参数说明**：设备ID

请求参数

表 1-642 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。

参数	是否必选	参数类型	描述
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-643 响应 Body 参数

参数	参数类型	描述
tunnels	Array of **TunnelInfo** objects	隧道信息列表。 数组长度：**0 - 100**

表 1-644 TunnelInfo

参数	参数类型	描述
tunnel_id	String	隧道ID
device_id	String	设备ID
create_time	String	隧道创建时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
closed_time	String	隧道更新时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
status	String	隧道状态 CLOSED | OPEN
source_connect_state	**ConnectState** object	访问端（console）状态
device_connect_state	**ConnectState** object	设备端状态

表 1-645 ConnectState

参数	参数类型	描述
last_update_time	String	隧道最近一次状态更新时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
status	String	客户端连接状态 CONNECTED | DISCONNECTED

请求示例

列表查询设备隧道。

```
GET https://{endpoint}/v5/iot/{project_id}/tunnels
```

响应示例

状态码： 200

查询设备所有隧道信息成功

```
{
  "tunnels" : [ {
    "tunnel_id" : "d144a524-1997-4b99-94bf-f27128da8a34",
    "device_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
    "create_time" : "20190303T081011Z",
    "closed_time" : "20190303T081011Z",
    "status" : "CLOSED",
    "source_connect_state" : {
      "last_update_time" : "20190303T081011Z",
      "status" : "CONNECTED"
    },
    "device_connect_state" : {
      "last_update_time" : "20190303T081011Z",
      "status" : "CONNECTED"
    }
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListDeviceTunnelsSolution {
```



```
    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListDeviceTunnelsRequest request = new ListDeviceTunnelsRequest();
        request.withDeviceId("<device_id>");
        try {
            ListDeviceTunnelsResponse response = client.listDeviceTunnels(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
```

```
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListDeviceTunnelsRequest()
        request.device_id = "<device_id>"
        response = client.list_device_tunnels(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListDeviceTunnelsRequest{}
    deviceIdRequest:= "<device_id>"
    request.DeviceId = &deviceIdRequest
    response, err := client.ListDeviceTunnels(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询设备所有隧道信息成功
400	输入参数不正确
401	Unauthorized
403	鉴权认证失败
500	Internal Server Error

错误码

请参见**错误码**。

1.4.19.3 查询设备隧道

功能介绍

用户可通过该接口查询某项目中的某个设备隧道，查看该设备隧道的信息与连接情况。应用服务器可调用此接口向平台查询设备隧道建立情况。

- 该API接口在基础版不支持。
- 具体应用可见“设备远程登录”功能，请参见**设备远程登录**。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/tunnels/{tunnel_id}

表 1-646 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
tunnel_id	是	String	隧道ID 最小长度：**1** 最大长度：**128**

请求参数

表 1-647 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-648 响应 Body 参数

参数	参数类型	描述
tunnel_id	String	隧道ID
device_id	String	设备ID
create_time	String	隧道创建时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
closed_time	String	隧道更新时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
status	String	隧道状态 CLOSED | OPEN
source_connect_state	**ConnectState** object	访问端（console）状态
device_connect_state	**ConnectState** object	设备端状态

表 1-649 ConnectState

参数	参数类型	描述
last_update_time	String	隧道最近一次状态更新时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
status	String	客户端连接状态 CONNECTED | DISCONNECTED

请求示例

查询设备隧道详情。

```
GET https://{endpoint}/v5/iot/{project_id}/tunnels/{tunnel_id}
```

响应示例

状态码： 200

OK

```
{
  "tunnel_id" : "d144a524-1997-4b99-94bf-f27128da8a34",
  "device_id" : "1a7ffc5c-d89c-44dd-8265-b1653d951ce0",
  "create_time" : "20190303T081011Z",
  "closed_time" : "20190303T081011Z",
  "status" : "CLOSED",
  "source_connect_state" : {
    "last_update_time" : "20190303T081011Z",
    "status" : "CONNECTED"
  },
  "device_connect_state" : {
    "last_update_time" : "20190303T081011Z",
    "status" : "CONNECTED"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDeviceTunnelSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
```



```
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDeviceTunnelRequest request = new ShowDeviceTunnelRequest();
        try {
            ShowDeviceTunnelResponse response = client.showDeviceTunnel(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
```

```
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDeviceTunnelRequest()
        response = client.show_device_tunnel(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDeviceTunnelRequest{}
    response, err := client.ShowDeviceTunnel(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	输入参数不正确
401	Unauthorized
403	鉴权认证失败
404	隧道不存在
500	Internal Server Error

错误码

请参见**错误码**。

1.4.19.4 关闭设备隧道

功能介绍

应用服务器可通过该接口关闭某个设备隧道。关闭后可以再次连接。

- 该API接口在基础版不支持。
- 具体应用可见“设备远程登录”功能，请参见**设备远程登录**。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/tunnels/{tunnel_id}

表 1-650 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
tunnel_id	是	String	隧道ID 最小长度：**1** 最大长度：**128**

请求参数

表 1-651 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

关闭设备隧道。

```
PUT https://{endpoint}/v5/iot/{project_id}/tunnels/{tunnel_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```



```
public class CloseDeviceTunnelSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CloseDeviceTunnelRequest request = new CloseDeviceTunnelRequest();
        try {
            CloseDeviceTunnelResponse response = client.closeDeviceTunnel(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CloseDeviceTunnelRequest()
        response = client.close_device_tunnel(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CloseDeviceTunnelRequest{}
    response, err := client.CloseDeviceTunnel(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	no content
400	输入参数不正确
401	Unauthorized
403	鉴权认证失败
404	隧道不存在
500	Internal Server Error

错误码

请参见**错误码**。

1.4.19.5 删除设备隧道

功能介绍

用户可通过该接口删除某个设备隧道。删除后该通道不存在，无法再次连接。

- 该API接口在基础版不支持。
- 具体应用可见“设备远程登录”功能，请参见**设备远程登录**。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/tunnels/{tunnel_id}

表 1-652 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
tunnel_id	是	String	隧道ID 最小长度：**1** 最大长度：**128**

请求参数

表 1-653 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除设备隧道。

```
DELELE https://{endpoint}/v5/iot/{project_id}/tunnels/{tunnel_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
```



```
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteDeviceTunnelSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteDeviceTunnelRequest request = new DeleteDeviceTunnelRequest();
        try {
            DeleteDeviceTunnelResponse response = client.deleteDeviceTunnel(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";
```

```
    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteDeviceTunnelRequest()
        response = client.delete_device_tunnel(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteDeviceTunnelRequest{}
    response, err := client.DeleteDeviceTunnel(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	no content
400	输入参数不正确
401	Unauthorized
403	鉴权认证失败
404	隧道不存在
500	Internal Server Error

错误码

请参见**错误码**。

1.4.20 数据流转积压策略管理

backlog policy of routing rule

1.4.20.1 新建数据流转积压策略

功能介绍

应用服务器可调用此接口在物联网平台创建数据流转积压策略。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/routing-rule/backlog-policy

表 1-654 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-655 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-656 请求 Body 参数

参数	是否必选	参数类型	描述
policy_name	否	String	**参数说明**：数据流转积压策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	否	String	**参数说明**：用户自定义的数据流转积压策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
backlog_size	否	Integer	**参数说明**：数据积压大小。单位为B（字节），取值范围为0~1073741823的整数，默认为1073741823（即1GB）。当backlog_size为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**1073741823** 缺省值：**1073741823**
backlog_time	否	Integer	**参数说明**：数据积压时间。单位为s（秒），取值范围为0~86399的整数，默认为86399（即1天）。当backlog_time为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**86399** 缺省值：**86399**

响应参数

状态码： 201

表 1-657 响应 Body 参数

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转积压策略id，用于唯一标识一个数据流转积压策略，在创建数据流转积压策略时由物联网平台分配获得。
policy_name	String	**参数说明**：数据流转积压策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
description	String	**参数说明**：用户自定义的数据流转积压策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
backlog_size	Integer	**参数说明**：数据积压大小。单位为B（字节），取值范围为0~1073741823的整数，默认为1073741823（即1GB）。当backlog_size为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**1073741823** 缺省值：**1073741823**
backlog_time	Integer	**参数说明**：数据积压时间。单位为s（秒），取值范围为0~86399的整数，默认为86399（即1天）。当backlog_time为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**86399** 缺省值：**86399**

请求示例

创建数据流转积压策略。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/backlog-policy

{
  "policy_name" : "rulename",
  "description" : "description",
  "backlog_size" : 100,
  "backlog_time" : 100
}
```

响应示例

状态码： 201

Created

```
{
  "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
  "policy_name" : "policyName",
  "description" : "description",
  "backlog_size" : 1073741823,
  "backlog_time" : 86399
}
```


SDK 代码示例

SDK代码示例如下。

Java

创建数据流转积压策略。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingBacklogPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingBacklogPolicyRequest request = new CreateRoutingBacklogPolicyRequest();
        AddBacklogPolicy body = new AddBacklogPolicy();
        body.withBacklogTime(100);
        body.withBacklogSize(100);
        body.withDescription("description");
        body.withPolicyName("rulename");
        request.withBody(body);
        try {
            CreateRoutingBacklogPolicyResponse response = client.createRoutingBacklogPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
```

```
    }
}
```

Python

创建数据流转积压策略。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingBacklogPolicyRequest()
        request.body = AddBacklogPolicy(
            backlog_time=100,
            backlog_size=100,
            description="description",
            policy_name="rulename"
        )
        response = client.create_routing_backlog_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建数据流转积压策略。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)
```

```
func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRoutingBacklogPolicyRequest{}
    backlogTimeAddBacklogPolicy:= int32(100)
    backlogSizeAddBacklogPolicy:= int32(100)
    descriptionAddBacklogPolicy:= "description"
    policyNameAddBacklogPolicy:= "rulename"
    request.Body = &model.AddBacklogPolicy{
        BacklogTime: &backlogTimeAddBacklogPolicy,
        BacklogSize: &backlogSizeAddBacklogPolicy,
        Description: &descriptionAddBacklogPolicy,
        PolicyName: &policyNameAddBacklogPolicy,
    }
    response, err := client.CreateRoutingBacklogPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.20.2 查询数据流转积压策略列表

功能介绍

应用服务器可调用此接口查询在物联网平台设置的数据流转积压策略列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/backlog-policy

表 1-658 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-659 Query 参数

参数	是否必选	参数类型	描述
policy_name	否	String	**参数说明**：数据流转积压策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。默认每页10条记录，最大设定每页50条记录。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。 - 限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-660 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-661 响应 Body 参数

参数	参数类型	描述
backlog_policies	Array of **BacklogPolicyInfo** objects	数据流转积压策略列表。 数组长度：**0 - 50**
count	Integer	满足查询条件的记录总数。 最小值：**0**
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

表 1-662 BacklogPolicyInfo

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转积压策略id，用于唯一标识一个数据流转积压策略，在创建数据流转积压策略时由物联网平台分配获得。

参数	参数类型	描述
policy_name	String	**参数说明**：数据流转积压策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	String	**参数说明**：用户自定义的数据流转积压策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
backlog_size	Integer	**参数说明**：数据积压大小。单位为B（字节），取值范围为0~1073741823的整数，默认为1073741823（即1GB）。当backlog_size为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**1073741823** 缺省值：**1073741823**
backlog_time	Integer	**参数说明**：数据积压时间。单位为s（秒），取值范围为0~86399的整数，默认为86399（即1天）。当backlog_time为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**86399** 缺省值：**86399**

请求示例

查询数据流转积压策略列表。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/backlog-policy
```

响应示例

状态码： 200

Ok

```
{
  "backlog_policies" : [ {
    "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
    "policy_name" : "policyName",
    "description" : "description",
    "backlog_size" : 1073741823,
    "backlog_time" : 86399
```



```
 } ],
 "count" : 10,
 "marker" : "5c90fa7d3c4e4405e8525079"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListRoutingBacklogPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListRoutingBacklogPolicyRequest request = new ListRoutingBacklogPolicyRequest();
        request.withPolicyName("<policy_name>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListRoutingBacklogPolicyResponse response = client.listRoutingBacklogPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
```

```
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListRoutingBacklogPolicyRequest()
        request.policy_name = "<policy_name>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_routing_backlog_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
```

```
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListRoutingBacklogPolicyRequest{}
    policyNameRequest:= "<policy_name>"
    request.PolicyName = &policyNameRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListRoutingBacklogPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.20.3 修改数据流转积压策略

功能介绍

应用服务器可调用此接口在物联网平台修改指定数据流转积压策略。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/routing-rule/backlog-policy/{policy_id}

表 1-663 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	**参数说明**：数据流转积压策略id，用于唯一标识一个数据流转积压策略，在创建数据流转积压策略时由物联网平台分配获得。

请求参数

表 1-664 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-665 请求 Body 参数

参数	是否必选	参数类型	描述
policy_name	否	String	**参数说明**：数据流转积压策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	否	String	**参数说明**：用户自定义的数据流转积压策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
backlog_size	否	Integer	**参数说明**：数据积压大小。单位为B（字节），取值范围为0~1073741823的整数，默认为1073741823（即1GB）。当backlog_size为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**1073741823** 缺省值：**1073741823**
backlog_time	否	Integer	**参数说明**：数据积压时间。单位为s（秒），取值范围为0~86399的整数，默认为86399（即1天）。当backlog_time为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**86399** 缺省值：**86399**

响应参数

状态码： 200

表 1-666 响应 Body 参数

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转积压策略id，用于唯一标识一个数据流转积压策略，在创建数据流转积压策略时由物联网平台分配获得。
policy_name	String	**参数说明**：数据流转积压策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	String	**参数说明**：用户自定义的数据流转积压策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
backlog_size	Integer	**参数说明**：数据积压大小。单位为B（字节），取值范围为0~1073741823的整数，默认为1073741823（即1GB）。当backlog_size为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**1073741823** 缺省值：**1073741823**
backlog_time	Integer	**参数说明**：数据积压时间。单位为s（秒），取值范围为0~86399的整数，默认为86399（即1天）。当backlog_time为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**86399** 缺省值：**86399**

请求示例

更新数据流转积压策略。

```
PUT https://{endpoint}/v5/iot/{project_id}/routing-rule/backlog-policy/{policy_id}

{
  "backlog_size" : 100,
  "backlog_time" : 100
}
```

响应示例

状态码： 200

Ok

```
{
  "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
  "policy_name" : "policyName",
  "description" : "description",
  "backlog_size" : 1073741823,
  "backlog_time" : 86399
}
```

SDK 代码示例

SDK代码示例如下。

Java

更新数据流转积压策略。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateRoutingBacklogPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateRoutingBacklogPolicyRequest request = new UpdateRoutingBacklogPolicyRequest();
        UpdateBacklogPolicy body = new UpdateBacklogPolicy();
        body.withBacklogTime(100);
        body.withBacklogSize(100);
        request.withBody(body);
        try {
```



```
            UpdateRoutingBacklogPolicyResponse response = client.updateRoutingBacklogPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

更新数据流转积压策略。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateRoutingBacklogPolicyRequest()
        request.body = UpdateBacklogPolicy(
            backlog_time=100,
            backlog_size=100
        )
        response = client.update_routing_backlog_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

更新数据流转积压策略。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateRoutingBacklogPolicyRequest{}
    backlogTimeUpdateBacklogPolicy:= int32(100)
    backlogSizeUpdateBacklogPolicy:= int32(100)
    request.Body = &model.UpdateBacklogPolicy{
        BacklogTime: &backlogTimeUpdateBacklogPolicy,
        BacklogSize: &backlogSizeUpdateBacklogPolicy,
    }
    response, err := client.UpdateRoutingBacklogPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Ok
400	Bad Request

状态码	描述
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.20.4 查询数据流转积压策略

功能介绍

应用服务器可调用此接口在物联网平台查询指定数据流转积压策略。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/backlog-policy/{policy_id}

表 1-667 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	**参数说明**：数据流转积压策略id，用于唯一标识一个数据流转积压策略，在创建数据流转积压策略时由物联网平台分配获得。

请求参数

表 1-668 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-669 响应 Body 参数

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转积压策略id，用于唯一标识一个数据流转积压策略，在创建数据流转积压策略时由物联网平台分配获得。
policy_name	String	**参数说明**：数据流转积压策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	String	**参数说明**：用户自定义的数据流转积压策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
backlog_size	Integer	**参数说明**：数据积压大小。单位为B（字节），取值范围为0~1073741823的整数，默认为1073741823（即1GB）。当backlog_size为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**1073741823** 缺省值：**1073741823**
backlog_time	Integer	**参数说明**：数据积压时间。单位为s（秒），取值范围为0~86399的整数，默认为86399（即1天）。当backlog_time为0时，表示不积压。若同时配置了backlog_size和backlog_time两个维度，则以最先达到阈值的维度为准。 最小值：**0** 最大值：**86399** 缺省值：**86399**

请求示例

查询指定数据流转积压策略。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/backlog-policy/{policy_id}
```

响应示例

状态码： 200

Ok

```
{
  "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
  "policy_name" : "policyName",
  "description" : "description",
  "backlog_size" : 1073741823,
  "backlog_time" : 86399
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
```



```
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowRoutingBacklogPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowRoutingBacklogPolicyRequest request = new ShowRoutingBacklogPolicyRequest();
        try {
            ShowRoutingBacklogPolicyResponse response = client.showRoutingBacklogPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
```

```
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowRoutingBacklogPolicyRequest()
        response = client.show_routing_backlog_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowRoutingBacklogPolicyRequest{}
    response, err := client.ShowRoutingBacklogPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.20.5 删除数据流转积压策略

功能介绍

应用服务器可调用此接口在物联网平台删除指定数据流转积压策略。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/routing-rule/backlog-policy/{policy_id}

表 1-670 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	**参数说明**：数据流转积压策略id，用于唯一标识一个数据流转积压策略，在创建数据流转积压策略时由物联网平台分配获得。

请求参数

表 1-671 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

无

请求示例

删除指定数据流转积压策略。

```
DELETE https://{endpoint}/v5/iot/{project_id}/routing-rule/backlog-policy/{policy_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```

```
public class DeleteRoutingBacklogPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteRoutingBacklogPolicyRequest request = new DeleteRoutingBacklogPolicyRequest();
        try {
            DeleteRoutingBacklogPolicyResponse response = client.deleteRoutingBacklogPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```



```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteRoutingBacklogPolicyRequest()
        response = client.delete_routing_backlog_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteRoutingBacklogPolicyRequest{}
    response, err := client.DeleteRoutingBacklogPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.21 数据流转流控策略管理

flowcontrol policy of routing rule

1.4.21.1 新建数据流转流控策略

功能介绍

应用服务器可调用此接口在物联网平台创建数据流转流控策略。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/routing-rule/flowcontrol-policy

表 1-672 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-673 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-674 请求 Body 参数

参数	是否必选	参数类型	描述
policy_name	否	String	**参数说明**：数据流转流控策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	否	String	**参数说明**：用户自定义的数据流转流控策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
scope	否	String	**参数说明**：流控策略作用域. **取值范围**： ● USER：租户级流控策略。 ● CHANNEL：转发通道级流控策略。 ● RULE：转发规则级流控策略。 ● ACTION：转发动作级流控策略。
scope_value	否	String	**参数说明**：流控策略作用域附加值。 scope取值为USER时，可不携带该字段，表示租户级流控。 scope取值为CHANNEL时，**取值范围**：HTTP_FORWARDING、DIS_FORWARDING、OBS_FORWARDING、AMQP_FORWARDING、DMS_KAFKA_FORWARDING。 scope取值为RULE时，该字段为对应的ruleId。 scope取值为ACTION时，该字段为对应的actionId。
limit	否	Integer	**参数说明**：数据转发流控大小。单位为tps，取值范围为1~1000的整数，默认为1000. 最小值：**1** 最大值：**1000** 缺省值：**1000**

响应参数

状态码： 201

表 1-675 响应 Body 参数

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转流控策略id，用于唯一标识一个数据流转流控策略，在创建数据流转流控策略时由物联网平台分配获得。

参数	参数类型	描述
policy_name	String	**参数说明**：数据流转流控策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	String	**参数说明**：用户自定义的数据流转流控策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
scope	String	**参数说明**：流控策略作用域. **取值范围**： ● USER：租户级流控策略。 ● CHANNEL：转发通道级流控策略。 ● RULE：转发规则级流控策略。 ● ACTION：转发动作级流控策略。
scope_value	String	**参数说明**：流控策略作用域附加值。 scope取值为USER时，可不携带该字段，表示租户级流控。 scope取值为CHANNEL时，**取值范围**：HTTP_FORWARDING、DIS_FORWARDING、OBS_FORWARDING、AMQP_FORWARDING、DMS_KAFKA_FORWARDING。 scope取值为RULE时，该字段为对应的ruleId。 scope取值为ACTION时，该字段为对应的actionId。
limit	Integer	**参数说明**：数据转发流控大小。单位为tps，取值范围为1~1000的整数，默认为1000. 最小值：**1** 最大值：**1000** 缺省值：**1000**

请求示例

- 创建数据流转流控策略-实例级流控。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy

{
  "policy_name" : "policy_name",
  "description" : "description",
  "scope" : "USER",
  "limit" : 100
}
```

- 创建数据流转流控策略-转发通道级流控。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy

{
  "policy_name" : "policy_name",
```



```
  "description" : "description",
  "scope" : "CHANNEL",
  "scope_value" : "HTTP_FORWARDING",
  "limit" : 100
}
```

- 创建数据流转流控策略-转发规则级流控。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy

{
  "policy_name" : "policy_name",
  "description" : "description",
  "scope" : "RULE",
  "scope_value" : "b0443335-2627-4ebe-bdef-276113646520",
  "limit" : 100
}
```

- 创建数据流转流控策略-转发动作级流控。

```
POST https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy

{
  "policy_name" : "policy_name",
  "description" : "description",
  "scope" : "ACTION",
  "scope_value" : "b0443335-2627-4ebe-bdef-276113646520",
  "limit" : 100
}
```

响应示例

状态码： 201

Created

```
{
  "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
  "policy_name" : "policyName",
  "description" : "description",
  "scope" : "CHANNEL",
  "scope_value" : "HTTP_FORWARDING",
  "limit" : 10
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建数据流转流控策略-实例级流控。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
```



```
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingFlowControlPolicyRequest request = new CreateRoutingFlowControlPolicyRequest();
        AddFlowControlPolicy body = new AddFlowControlPolicy();
        body.withLimit(100);
        body.withScope("USER");
        body.withDescription("description");
        body.withPolicyName("policy_name");
        request.withBody(body);
        try {
            CreateRoutingFlowControlPolicyResponse response = client.createRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建数据流转流控策略-转发通道级流控。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
```

```
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingFlowControlPolicyRequest request = new
CreateRoutingFlowControlPolicyRequest();
        AddFlowControlPolicy body = new AddFlowControlPolicy();
        body.withLimit(100);
        body.withScopeValue("HTTP_FORWARDING");
        body.withScope("CHANNEL");
        body.withDescription("description");
        body.withPolicyName("policy_name");
        request.withBody(body);
        try {
            CreateRoutingFlowControlPolicyResponse response =
client.createRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建数据流转流控策略-转发规则级流控。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
```

```
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingFlowControlPolicyRequest request = new
CreateRoutingFlowControlPolicyRequest();
        AddFlowControlPolicy body = new AddFlowControlPolicy();
        body.withLimit(100);
        body.withScopeValue("b0443335-2627-4ebe-bdef-276113646520");
        body.withScope("RULE");
        body.withDescription("description");
        body.withPolicyName("policy_name");
        request.withBody(body);
        try {
            CreateRoutingFlowControlPolicyResponse response =
client.createRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建数据流转流控策略-转发动作级流控。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
```

```
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateRoutingFlowControlPolicyRequest request = new
CreateRoutingFlowControlPolicyRequest();
        AddFlowControlPolicy body = new AddFlowControlPolicy();
        body.withLimit(100);
        body.withScopeValue("b0443335-2627-4ebe-bdef-276113646520");
        body.withScope("ACTION");
        body.withDescription("description");
        body.withPolicyName("policy_name");
        request.withBody(body);
        try {
            CreateRoutingFlowControlPolicyResponse response =
client.createRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建数据流转流控策略-实例级流控。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
```

```
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingFlowControlPolicyRequest()
        request.body = AddFlowControlPolicy(
            limit=100,
            scope="USER",
            description="description",
            policy_name="policy_name"
        )
        response = client.create_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建数据流转流控策略-转发通道级流控。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingFlowControlPolicyRequest()
        request.body = AddFlowControlPolicy(
            limit=100,
            scope_value="HTTP_FORWARDING",
            scope="CHANNEL",
            description="description",
            policy_name="policy_name"
        )
        response = client.create_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
```



```
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建数据流转流控策略-转发规则级流控。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingFlowControlPolicyRequest()
        request.body = AddFlowControlPolicy(
            limit=100,
            scope_value="b0443335-2627-4ebe-bdef-276113646520",
            scope="RULE",
            description="description",
            policy_name="policy_name"
        )
        response = client.create_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建数据流转流控策略-转发动作级流控。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
```

```
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateRoutingFlowControlPolicyRequest()
        request.body = AddFlowControlPolicy(
            limit=100,
            scope_value="b0443335-2627-4ebe-bdef-276113646520",
            scope="ACTION",
            description="description",
            policy_name="policy_name"
        )
        response = client.create_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建数据流转流控策略-实例级流控。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
```

```
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRoutingFlowControlPolicyRequest{}
    limitAddFlowControlPolicy:= int32(100)
    scopeAddFlowControlPolicy:= "USER"
    descriptionAddFlowControlPolicy:= "description"
    policyNameAddFlowControlPolicy:= "policy_name"
    request.Body = &model.AddFlowControlPolicy{
        Limit: &limitAddFlowControlPolicy,
        Scope: &scopeAddFlowControlPolicy,
        Description: &descriptionAddFlowControlPolicy,
        PolicyName: &policyNameAddFlowControlPolicy,
    }
    response, err := client.CreateRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建数据流转流控策略-转发通道级流控。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRoutingFlowControlPolicyRequest{}
    limitAddFlowControlPolicy:= int32(100)
    scopeValueAddFlowControlPolicy:= "HTTP_FORWARDING"
    scopeAddFlowControlPolicy:= "CHANNEL"
    descriptionAddFlowControlPolicy:= "description"
    policyNameAddFlowControlPolicy:= "policy_name"
    request.Body = &model.AddFlowControlPolicy{
        Limit: &limitAddFlowControlPolicy,
```

```
        ScopeValue: &scopeValueAddFlowControlPolicy,
        Scope: &scopeAddFlowControlPolicy,
        Description: &descriptionAddFlowControlPolicy,
        PolicyName: &policyNameAddFlowControlPolicy,
    }
    response, err := client.CreateRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建数据流转流控策略-转发规则级流控。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRoutingFlowControlPolicyRequest{}
    limitAddFlowControlPolicy:= int32(100)
    scopeValueAddFlowControlPolicy:= "b0443335-2627-4ebe-bdef-276113646520"
    scopeAddFlowControlPolicy:= "RULE"
    descriptionAddFlowControlPolicy:= "description"
    policyNameAddFlowControlPolicy:= "policy_name"
    request.Body = &model.AddFlowControlPolicy{
        Limit: &limitAddFlowControlPolicy,
        ScopeValue: &scopeValueAddFlowControlPolicy,
        Scope: &scopeAddFlowControlPolicy,
        Description: &descriptionAddFlowControlPolicy,
        PolicyName: &policyNameAddFlowControlPolicy,
    }
    response, err := client.CreateRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```



```
    }
}
```

- 创建数据流转流控策略-转发动作级流控。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateRoutingFlowControlPolicyRequest{}
    limitAddFlowControlPolicy:= int32(100)
    scopeValueAddFlowControlPolicy:= "b0443335-2627-4ebe-bdef-276113646520"
    scopeAddFlowControlPolicy:= "ACTION"
    descriptionAddFlowControlPolicy:= "description"
    policyNameAddFlowControlPolicy:= "policy_name"
    request.Body = &model.AddFlowControlPolicy{
        Limit: &limitAddFlowControlPolicy,
        ScopeValue: &scopeValueAddFlowControlPolicy,
        Scope: &scopeAddFlowControlPolicy,
        Description: &descriptionAddFlowControlPolicy,
        PolicyName: &policyNameAddFlowControlPolicy,
    }
    response, err := client.CreateRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.21.2 查询数据流转流控策略列表

功能介绍

应用服务器可调用此接口查询在物联网平台设置的数据流转流控策略列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/flowcontrol-policy

表 1-676 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-677 Query 参数

参数	是否必选	参数类型	描述
scope	否	String	**参数说明**：流控策略作用域。不携带该参数时，查询所有作用域流控策略；取值USER时，查询租户级流控策略；取值为CHANNEL时，查询转发通道级流控策略；取值为RULE时，查询转发规则级流控策略；取值为ACTION时，查询转发动作级流控策略。
scope_value	否	String	**参数说明**：流控策略作用域附加值。 不携带scope参数或scope参数取值为USER时，可不携带该字段，查询租户级流控策略。scope参数取值为CHANNEL时，不携带该字段表示查询所有转发通道级流控策略，携带该字段表示查询该字段取值对应转发通道的流控策略。**取值范围**：HTTP_FORWARDING、DIS_FORWARDING、OBS_FORWARDING、AMQP_FORWARDING、DMS_KAFKA_FORWARDING。scope参数为RULE时，不携带该字段表示查询所有转发规则级流控策略，携带该字段表示查询该字段取值对应转发规则的流控策略。 scope参数为ACTION时，不携带该字段表示查询所有转发动作级流控策略，携带该字段表示查询该字段取值对应转发动作的流控策略。
policy_name	否	String	**参数说明**：数据流转流控策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**

参数	是否必选	参数类型	描述
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。默认每页10条记录，最大设定每页50条记录。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。 - 限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-678 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-679 响应 Body 参数

参数	参数类型	描述
flowcontrol_policies	Array of **FlowControlPolicyInfo** objects	数据流转流控策略列表。 数组长度：**0 - 50**
count	Integer	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

表 1-680 FlowControlPolicyInfo

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转流控策略id，用于唯一标识一个数据流转流控策略，在创建数据流转流控策略时由物联网平台分配获得。

参数	参数类型	描述
policy_name	String	**参数说明**：数据流转流控策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	String	**参数说明**：用户自定义的数据流转流控策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
scope	String	**参数说明**：流控策略作用域. **取值范围**： ● USER：租户级流控策略。 ● CHANNEL：转发通道级流控策略。 ● RULE：转发规则级流控策略。 ● ACTION：转发动作级流控策略。
scope_value	String	**参数说明**：流控策略作用域附加值。 scope取值为USER时，可不携带该字段，表示租户级流控。 scope取值为CHANNEL时，**取值范围**：HTTP_FORWARDING、DIS_FORWARDING、OBS_FORWARDING、AMQP_FORWARDING、DMS_KAFKA_FORWARDING。 scope取值为RULE时，该字段为对应的ruleId。 scope取值为ACTION时，该字段为对应的actionId。
limit	Integer	**参数说明**：数据转发流控大小。单位为tps，取值范围为1~1000的整数，默认为1000. 最小值：**1** 最大值：**1000** 缺省值：**1000**

请求示例

查询数据流转流控策略列表。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy
```

响应示例

状态码： 200

Ok

```
{
  "flowcontrol_policies" : [ {
    "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
    "policy_name" : "policyName",
```



```
    "description" : "description",
    "scope" : "CHANNEL",
    "scope_value" : "HTTP_FORWARDING",
    "limit" : 10
  } ],
  "count" : 10,
  "marker" : "5c90fa7d3c4e4405e8525079"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListRoutingFlowControlPolicyRequest request = new ListRoutingFlowControlPolicyRequest();
        request.withScope("<scope>");
        request.withScopeValue("<scope_value>");
        request.withPolicyName("<policy_name>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListRoutingFlowControlPolicyResponse response = client.listRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
```

```
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListRoutingFlowControlPolicyRequest()
        request.scope = "<scope>"
        request.scope_value = "<scope_value>"
        request.policy_name = "<policy_name>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
```

```
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListRoutingFlowControlPolicyRequest{}
    scopeRequest:= "<scope>"
    request.Scope = &scopeRequest
    scopeValueRequest:= "<scope_value>"
    request.ScopeValue = &scopeValueRequest
    policyNameRequest:= "<policy_name>"
    request.PolicyName = &policyNameRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden

状态码	描述
500	Internal Server Error

错误码

请参见**错误码**。

1.4.21.3 修改数据流转流控策略

功能介绍

应用服务器可调用此接口在物联网平台修改指定数据流转流控策略。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/routing-rule/flowcontrol-policy/{policy_id}

表 1-681 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	**参数说明**：数据流转流控策略id，用于唯一标识一个数据流转流控策略，在创建数据流转流控策略时由物联网平台分配获得。

请求参数

表 1-682 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。

参数	是否必选	参数类型	描述
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

表 1-683 请求 Body 参数

参数	是否必选	参数类型	描述
policy_name	否	String	**参数说明**：数据流转流控策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	否	String	**参数说明**：用户自定义的数据流转流控策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
limit	否	Integer	**参数说明**：数据转发流控大小。单位为tps，取值范围为1~1000的整数，默认为1000. 最小值：**1** 最大值：**1000** 缺省值：**1000**

响应参数

状态码： 200

表 1-684 响应 Body 参数

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转流控策略id，用于唯一标识一个数据流转流控策略，在创建数据流转流控策略时由物联网平台分配获得。
policy_name	String	**参数说明**：数据流转流控策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	String	**参数说明**：用户自定义的数据流转流控策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
scope	String	**参数说明**：流控策略作用域. **取值范围**： ● USER：租户级流控策略。 ● CHANNEL：转发通道级流控策略。 ● RULE：转发规则级流控策略。 ● ACTION：转发动作级流控策略。
scope_value	String	**参数说明**：流控策略作用域附加值。 scope取值为USER时，可不携带该字段，表示租户级流控。scope取值为CHANNEL时，**取值范围**：HTTP_FORWARDING、DIS_FORWARDING、OBS_FORWARDING、AMQP_FORWARDING、DMS_KAFKA_FORWARDING。 scope取值为RULE时，该字段为对应的ruleId。 scope取值为ACTION时，该字段为对应的actionId。
limit	Integer	**参数说明**：数据转发流控大小。单位为tps，取值范围为1~1000的整数，默认为1000. 最小值：**1** 最大值：**1000** 缺省值：**1000**

请求示例

修改数据流转流控策略。

```
PUT https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy/{policy_id}

{
  "policy_name" : "policyName",
  "description" : "description",
  "limit" : 1000
}
```


响应示例

状态码： 200

Ok

```
{
  "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
  "policy_name" : "policyName",
  "description" : "description",
  "scope" : "CHANNEL",
  "scope_value" : "HTTP_FORWARDING",
  "limit" : 10
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改数据流转流控策略。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateRoutingFlowControlPolicyRequest request = new UpdateRoutingFlowControlPolicyRequest();
        UpdateFlowControlPolicy body = new UpdateFlowControlPolicy();
        body.withLimit(1000);
        body.withDescription("description");
        body.withPolicyName("policyName");
```



```
        request.withBody(body);
        try {
            UpdateRoutingFlowControlPolicyResponse response =
client.updateRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改数据流转流控策略。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateRoutingFlowControlPolicyRequest()
        request.body = UpdateFlowControlPolicy(
            limit=1000,
            description="description",
            policy_name="policyName"
        )
        response = client.update_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改数据流转流控策略。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateRoutingFlowControlPolicyRequest{}
    limitUpdateFlowControlPolicy:= int32(1000)
    descriptionUpdateFlowControlPolicy:= "description"
    policyNameUpdateFlowControlPolicy:= "policyName"
    request.Body = &model.UpdateFlowControlPolicy{
        Limit: &limitUpdateFlowControlPolicy,
        Description: &descriptionUpdateFlowControlPolicy,
        PolicyName: &policyNameUpdateFlowControlPolicy,
    }
    response, err := client.UpdateRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.21.4 查询数据流转流控策略

功能介绍

应用服务器可调用此接口在物联网平台查询指定数据流转流控策略。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/routing-rule/flowcontrol-policy/{policy_id}

表 1-685 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	**参数说明**：数据流转流控策略id，用于唯一标识一个数据流转流控策略，在创建数据流转流控策略时由物联网平台分配获得。

请求参数

表 1-686 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

状态码： 200

表 1-687 响应 Body 参数

参数	参数类型	描述
policy_id	String	**参数说明**：数据流转流控策略id，用于唯一标识一个数据流转流控策略，在创建数据流转流控策略时由物联网平台分配获得。
policy_name	String	**参数说明**：数据流转流控策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**
description	String	**参数说明**：用户自定义的数据流转流控策略描述。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**256**

参数	参数类型	描述
scope	String	**参数说明**：流控策略作用域. **取值范围**： ● USER：租户级流控策略。 ● CHANNEL：转发通道级流控策略。 ● RULE：转发规则级流控策略。 ● ACTION：转发动作级流控策略。
scope_value	String	**参数说明**：流控策略作用域附加值。 scope取值为USER时，可不携带该字段，表示租户级流控。 scope取值为CHANNEL时，**取值范围**：HTTP_FORWARDING、DIS_FORWARDING、OBS_FORWARDING、AMQP_FORWARDING、DMS_KAFKA_FORWARDING。 scope取值为RULE时，该字段为对应的ruleId。 scope取值为ACTION时，该字段为对应的actionId。
limit	Integer	**参数说明**：数据转发流控大小。单位为tps，取值范围为1~1000的整数，默认为1000. 最小值：**1** 最大值：**1000** 缺省值：**1000**

请求示例

查询指定数据流转流控策略。

```
GET https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy/{policy_id}
```

响应示例

状态码： 200

Ok

```
{
  "policy_id" : "adadd5cb-6383-4b5b-a65c-f8c92fdf3c34",
  "policy_name" : "policyName",
  "description" : "description",
  "scope" : "CHANNEL",
  "scope_value" : "HTTP_FORWARDING",
  "limit" : 10
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
```



```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowRoutingFlowControlPolicyRequest request = new ShowRoutingFlowControlPolicyRequest();
        try {
            ShowRoutingFlowControlPolicyResponse response = client.showRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
```

```
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowRoutingFlowControlPolicyRequest()
        response = client.show_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowRoutingFlowControlPolicyRequest{}
    response, err := client.ShowRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
```



```
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.21.5 删除数据流转流控策略

功能介绍

应用服务器可调用此接口在物联网平台删除指定数据流转流控策略。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/routing-rule/flowcontrol-policy/{policy_id}

表 1-688 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

参数	是否必选	参数类型	描述
policy_id	是	String	**参数说明**：数据流转流控策略id，用于唯一标识一个数据流转流控策略，在创建数据流转流控策略时由物联网平台分配获得。

请求参数

表 1-689 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务**获取IAM用户Token**接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，一般华为云租户无需携带该参数，仅在物理多租场景下从管理面访问API时需要携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID。

响应参数

无

请求示例

删除指定数据流转流控策略。

```
DELETE https://{endpoint}/v5/iot/{project_id}/routing-rule/flowcontrol-policy/{policy_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteRoutingFlowControlPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteRoutingFlowControlPolicyRequest request = new DeleteRoutingFlowControlPolicyRequest();
        try {
            DeleteRoutingFlowControlPolicyResponse response = client.deleteRoutingFlowControlPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
```

```
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteRoutingFlowControlPolicyRequest()
        response = client.delete_routing_flow_control_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
```

```
        WithCredential(auth).
        Build())

    request := &model.DeleteRoutingFlowControlPolicyRequest{}
    response, err := client.DeleteRoutingFlowControlPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.22 设备代理

1.4.22.1 创建设备代理

功能介绍

应用服务器可调用此接口在物联网平台创建一个动态设备代理规则，用于子设备自主选择网关设备上线和上报消息，即代理组下的任意网关下的子设备均可以通过代理组里其他设备上线(**网关更新子设备状态**)然后进行数据上报(**网关批量设备属性上报**)。

- 单实例最多可以配置10个设备代理
- 单账号调用该接口的 TPS 限制最大为1/S(每秒1次请求数)

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-proxies

表 1-690 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-691 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-692 请求 Body 参数

参数	是否必选	参数类型	描述
proxy_name	是	String	**参数说明**：设备代理名称 最小长度：**1** 最大长度：**64**

参数	是否必选	参数类型	描述
proxy_devices	是	Array of strings	**参数说明**：代理设备列表，列表内所有设备共享网关权限，即列表内任意一个网关下的子设备可以通过组里任意一个网关上线然后进行数据上报。 **取值范围**：列表内填写设备id，列表内最少有2个设备id，最多有10个设备id，设备id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，建议不少于4个字符。
effective_time_range	是	EffectiveTimeRange object	**参数说明**：设备代理规则有效期
app_id	是	String	**参数说明**：资源空间ID。携带该参数指定创建的设备归属到哪个资源空间下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。

表 1-693 EffectiveTimeRange

参数	是否必选	参数类型	描述
start_time	否	String	设备代理开始生效的时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'
end_time	否	String	设备代理失效的时间，必须大于start_time，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'

响应参数

状态码： 201

表 1-694 响应 Body 参数

参数	参数类型	描述
proxy_id	String	**参数说明**：设备代理ID。用来唯一标识一个代理规则
proxy_name	String	**参数说明**：设备代理名称

参数	参数类型	描述
proxy_devices	Array of strings	**参数说明**：代理设备组，组内所有设备共享网关权限，即组内任意一个网关下的子设备可以通过组里任意一个网关上线然后进行数据上报。
effective_time_range	**EffectiveTimeRangeResponseDTO** object	**参数说明**：设备代理规则有效期
app_id	String	**参数说明**：资源空间ID。

表 1-695 EffectiveTimeRangeResponseDTO

参数	参数类型	描述
start_time	String	设备代理开始生效的时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'
end_time	String	设备代理失效的时间，必须大于start_time，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'

请求示例

创建设备代理

```
POST https://{endpoint}/v5/iot/{project_id}/device-proxies

{
  "proxy_name" : "testAPP01",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "proxy_devices" : [ "d4922d8a-6c8e-4396-852c-164aefa6638f",
"d4922d8a-6c8e-4396-852c-164aefa6638g" ],
  "effective_time_range" : {
    "start_time" : "20200812T121212Z",
    "end_time" : "20210812T121212Z"
  }
}
```

响应示例

状态码： 201

Created

```
{
  "proxy_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
  "proxy_name" : "testAPP01",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "proxy_devices" : [ "d4922d8a-6c8e-4396-852c-164aefa6638f",
"d4922d8a-6c8e-4396-852c-164aefa6638g" ],
  "effective_time_range" : {
    "start_time" : "20200812T121212Z",
    "end_time" : "20210812T121212Z"
  }
}
```


状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.22.2 查询设备代理列表

功能介绍

应用服务器可调用此接口查询物联网平台中的设备代理列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-proxies

表 1-696 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-697 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的设备代理列表，不携带该参数则会查询该用户下所有设备代理。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
proxy_name	否	String	**参数说明**：设备代理名称。 **取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**64**
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-698 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-699 响应 Body 参数

参数	参数类型	描述
device_proxies	Array of **QueryDeviceProxySimplify** objects	代理设备列表
page	**Page** object	查询结果的分页信息。

表 1-700 QueryDeviceProxySimplify

参数	参数类型	描述
proxy_id	String	**参数说明**：设备代理ID。用来唯一标识一个代理规则
proxy_name	String	**参数说明**：设备代理名称
effective_time_range	**EffectiveTimeRangeResponseDTO** object	**参数说明**：规则有效期
app_id	String	**参数说明**：资源空间ID。

表 1-701 EffectiveTimeRangeResponseDTO

参数	参数类型	描述
start_time	String	设备代理开始生效的时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'
end_time	String	设备代理失效的时间，必须大于start_time，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'

表 1-702 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

```
GET https://{endpoint}/v5/iot/{project_id}/device-proxies
```

响应示例

状态码： 200

OK

```
{
  "device_proxies" : [ {
    "proxy_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
    "proxy_name" : "testProxyName",
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "effective_time_range" : {
      "start_time" : "20200812T121212Z",
      "end_time" : "20210812T121212Z"
    }
  } ],
  "page" : {
    "count" : 1,
    "marker" : "66178add3b98944277731d0a"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListDeviceProxiesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
```

```
        ListDeviceProxiesRequest request = new ListDeviceProxiesRequest();
        request.withAppId("<app_id>");
        request.withProxyName("<proxy_name>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListDeviceProxiesResponse response = client.listDeviceProxies(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListDeviceProxiesRequest()
        request.app_id = "<app_id>"
        request.proxy_name = "<proxy_name>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_device_proxies(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListDeviceProxiesRequest{}
    appIdRequest:= "<app_id>"
    request.AppId = &appIdRequest
    proxyNameRequest:= "<proxy_name>"
    request.ProxyName = &proxyNameRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListDeviceProxies(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.22.3 查询设备代理详情

功能介绍

应用服务器可调用此接口查询物联网平台中指定设备代理的详细信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-proxies/{proxy_id}

表 1-703 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
proxy_id	是	String	**参数说明**：设备代理ID，用于唯一标识一个设备代理。在注册设备代理时由物联网平台分配获得。

请求参数

表 1-704 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-705 响应 Body 参数

参数	参数类型	描述
proxy_id	String	**参数说明**：设备代理ID。用来唯一标识一个代理规则
proxy_name	String	**参数说明**：设备代理名称
proxy_devices	Array of strings	**参数说明**：代理设备组，组内所有设备共享网关权限，即组内任意一个网关下的子设备可以通过组里任意一个网关上线然后进行数据上报。
effective_time_range	**EffectiveTimeRangeResponseDTO** object	**参数说明**：设备代理规则有效期
app_id	String	**参数说明**：资源空间ID。

表 1-706 EffectiveTimeRangeResponseDTO

参数	参数类型	描述
start_time	String	设备代理开始生效的时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'
end_time	String	设备代理失效的时间，必须大于start_time，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'

请求示例

```
GET https://{endpoint}/v5/iot/{project_id}/device-proxies/{proxy_id}
```

响应示例

状态码： 200

OK

```
{
  "proxy_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
  "proxy_name" : "testAPP01",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "proxy_devices" : [ "d4922d8a-6c8e-4396-852c-164aefa6638f",
"d4922d8a-6c8e-4396-852c-164aefa6638g" ],
  "effective_time_range" : {
    "start_time" : "20200812T121212Z",
    "end_time" : "20210812T121212Z"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDeviceProxySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running
this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
```



```
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDeviceProxyRequest request = new ShowDeviceProxyRequest();
        try {
            ShowDeviceProxyResponse response = client.showDeviceProxy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDeviceProxyRequest()
        response = client.show_device_proxy(request)
        print(response)
```



```
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDeviceProxyRequest{}
    response, err := client.ShowDeviceProxy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK

状态码	描述
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.22.4 修改设备代理

功能介绍

应用服务器可调用此接口修改物联网平台中指定设备代理的基本信息。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/device-proxies/{proxy_id}

表 1-707 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
proxy_id	是	String	**参数说明**：设备代理ID，用于唯一标识一个设备代理。在注册设备代理时由物联网平台分配获得。

请求参数

表 1-708 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-709 请求 Body 参数

参数	是否必选	参数类型	描述
proxy_name	否	String	**参数说明**：设备代理名称 最小长度：**1** 最大长度：**64**
proxy_devices	否	Array of strings	**参数说明**：代理设备列表，列表内所有设备共享网关权限，即列表内任意一个网关下的子设备可以通过组里任意一个网关上线然后进行数据上报。 **取值范围**：列表内填写设备id，列表内最少有2个设备id，最多有10个设备id，设备id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，建议不少于4个字符。
effective_time_range	否	**EffectiveTimeRange** object	**参数说明**：规则有效期。

表 1-710 EffectiveTimeRange

参数	是否必选	参数类型	描述
start_time	否	String	设备代理开始生效的时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'
end_time	否	String	设备代理失效的时间，必须大于start_time，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'

响应参数

状态码： 200

表 1-711 响应 Body 参数

参数	参数类型	描述
proxy_id	String	**参数说明**：设备代理ID。用来唯一标识一个代理规则
proxy_name	String	**参数说明**：设备代理名称
proxy_devices	Array of strings	**参数说明**：代理设备组，组内所有设备共享网关权限，即组内任意一个网关下的子设备可以通过组里任意一个网关上线然后进行数据上报。
effective_time_range	**EffectiveTimeRangeResponseDTO** object	**参数说明**：设备代理规则有效期
app_id	String	**参数说明**：资源空间ID。

表 1-712 EffectiveTimeRangeResponseDTO

参数	参数类型	描述
start_time	String	设备代理开始生效的时间，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'
end_time	String	设备代理失效的时间，必须大于start_time，使用UTC时区，格式：yyyyMMdd'T'HHmmss'Z'

请求示例

修改设备代理

```
PUT https://{endpoint}/v5/iot/{project_id}/device-proxies/{proxy_id}
```



```
{
  "proxy_name" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "proxy_devices" : [ "d4922d8a-6c8e-4396-852c-164aefa6638f",
"d4922d8a-6c8e-4396-852c-164aefa6638g" ],
  "effective_time_range" : {
    "start_time" : "20200812T121212Z",
    "end_time" : "20200812T121212Z"
  }
}
```

响应示例

状态码： 200

OK

```
{
  "proxy_id" : "04ed32dc1b0025b52fe3c01a27c2babc",
  "proxy_name" : "testAPP01",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "proxy_devices" : [ "d4922d8a-6c8e-4396-852c-164aefa6638f",
"d4922d8a-6c8e-4396-852c-164aefa6638g" ],
  "effective_time_range" : {
    "start_time" : "20200812T121212Z",
    "end_time" : "20210812T121212Z"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改设备代理

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateDeviceProxySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running
this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
```

```
            .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
            .withAk(ak)
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        UpdateDeviceProxyRequest request = new UpdateDeviceProxyRequest();
        UpdateDeviceProxy body = new UpdateDeviceProxy();
        EffectiveTimeRange effectiveTimeRangebody = new EffectiveTimeRange();
        effectiveTimeRangebody.withStartTime("20200812T121212Z")
            .withEndTime("20200812T121212Z");
        List<String> listbodyProxyDevices = new ArrayList<>();
        listbodyProxyDevices.add("d4922d8a-6c8e-4396-852c-164aefa6638f");
        listbodyProxyDevices.add("d4922d8a-6c8e-4396-852c-164aefa6638g");
        body.withEffectiveTimeRange(effectiveTimeRangebody);
        body.withProxyDevices(listbodyProxyDevices);
        body.withProxyName("jeQDJQZltU8iKgFFoW060F5SGZka");
        request.withBody(body);
        try {
            UpdateDeviceProxyResponse response = client.updateDeviceProxy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改设备代理

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
```

```
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateDeviceProxyRequest()
        effectiveTimeRangebody = EffectiveTimeRange(
            start_time="20200812T121212Z",
            end_time="20200812T121212Z"
        )
        listProxyDevicesbody = [
            "d4922d8a-6c8e-4396-852c-164aefa6638f",
            "d4922d8a-6c8e-4396-852c-164aefa6638g"
        ]
        request.body = UpdateDeviceProxy(
            effective_time_range=effectiveTimeRangebody,
            proxy_devices=listProxyDevicesbody,
            proxy_name="jeQDJQZltU8iKgFFoW060F5SGZka"
        )
        response = client.update_device_proxy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改设备代理

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())
```



```
    request := &model.UpdateDeviceProxyRequest{}
    startTimeEffectiveTimeRange:= "20200812T121212Z"
    endTimeEffectiveTimeRange:= "20200812T121212Z"
    effectiveTimeRangebody := &model.EffectiveTimeRange{
        StartTime: &startTimeEffectiveTimeRange,
        EndTime: &endTimeEffectiveTimeRange,
    }
    var listProxyDevicesbody = []string{
        "d4922d8a-6c8e-4396-852c-164aefa6638f",
        "d4922d8a-6c8e-4396-852c-164aefa6638g",
    }
    proxyNameUpdateDeviceProxy:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    request.Body = &model.UpdateDeviceProxy{
        EffectiveTimeRange: effectiveTimeRangebody,
        ProxyDevices: &listProxyDevicesbody,
        ProxyName: &proxyNameUpdateDeviceProxy,
    }
    response, err := client.UpdateDeviceProxy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.22.5 删除设备代理

功能介绍

应用服务器可调用此接口在物联网平台上删除指定设备代理。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/device-proxies/{proxy_id}

表 1-713 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
proxy_id	是	String	**参数说明**：设备代理ID，用于唯一标识一个设备代理。在注册设备代理时由物联网平台分配获得。

请求参数

表 1-714 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

```
DELETE https://{endpoint}/v5/iot/{project_id}/device-proxies/{proxy_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteDeviceProxySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteDeviceProxyRequest request = new DeleteDeviceProxyRequest();
        try {
            DeleteDeviceProxyResponse response = client.deleteDeviceProxy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
```



```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteDeviceProxyRequest()
        response = client.delete_device_proxy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteDeviceProxyRequest{}
    response, err := client.DeleteDeviceProxy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.23 网桥管理

1.4.23.1 创建网桥

功能介绍

应用服务器可调用此接口在物联网平台创建一个网桥，仅在创建后的网桥才可以接入物联网平台。

- 一个实例最多支持20个网桥。
- 仅**标准版实例、企业版实例**支持该接口调用，基础版不支持。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/bridges

表 1-715 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-716 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-717 请求 Body 参数

参数	是否必选	参数类型	描述
bridge_name	是	String	网桥名称。**取值范围**：长度不超过64，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。 最小长度：**1** 最大长度：**64**

参数	是否必选	参数类型	描述
bridge_id	否	String	网桥ID。**取值范围**：长度不超过36，只允许字母、数字、_-字符的组合。

响应参数

状态码： 201

表 1-718 响应 Body 参数

参数	参数类型	描述
bridge_id	String	网桥ID，用于唯一标识一个网桥。在注册网桥时直接指定，或者由物联网平台分配获得。
bridge_name	String	网桥名称。
auth_info	**BridgeAuthInfo** object	网桥鉴权信息
create_time	String	在物联网平台注册网桥的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-719 BridgeAuthInfo

参数	参数类型	描述
auth_type	String	鉴权类型。当前支持密钥认证接入(SECRET)。使用密钥认证接入方式(SECRET)填写secret字段，不填写auth_type默认为密钥认证接入方式(SECRET)。
secret	String	网桥密钥，认证类型使用密钥认证接入(SECRET)可填写该字段。

请求示例

```
POST https://{endpoint}/v5/iot/{project_id}/bridges

{
  "bridge_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "bridge_name" : "dianabridge"
}
```

响应示例

状态码： 201

Created


```
{
  "bridge_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "bridge_name" : "dianabridge",
  "auth_info" : {
    "auth_type" : "SECRET",
    "secret" : "3b935******dc3c"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class AddBridgeSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        AddBridgeRequest request = new AddBridgeRequest();
        AddBridge body = new AddBridge();
        body.withBridgeId("d4922d8a-6c8e-4396-852c-164aefa6638f");
        body.withBridgeName("dianabridge");
        request.withBody(body);
        try {
            AddBridgeResponse response = client.addBridge(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
```



```
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = AddBridgeRequest()
        request.body = AddBridge(
            bridge_id="d4922d8a-6c8e-4396-852c-164aefa6638f",
            bridge_name="dianabridge"
        )
        response = client.add_bridge(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
```

```
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.AddBridgeRequest{}
    bridgeIdAddBridge:= "d4922d8a-6c8e-4396-852c-164aefa6638f"
    request.Body = &model.AddBridge{
        BridgeId: &bridgeIdAddBridge,
        BridgeName: "dianabridge",
    }
    response, err := client.AddBridge(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.23.2 查询网桥列表

功能介绍

应用服务器可调用此接口在物联网平台查询网桥列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/bridges

表 1-720 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-721 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**

参数	是否必选	参数类型	描述
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-722 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-723 响应 Body 参数

参数	参数类型	描述
bridges	Array of **BridgeResponse** objects	网桥列表。
page	**Page** object	查询结果的分页信息。

表 1-724 BridgeResponse

参数	参数类型	描述
bridge_id	String	网桥ID
bridge_name	String	网桥名称。
status	String	网桥状态。 • ONLINE：网桥在线。 • OFFLINE：网桥离线。

表 1-725 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

```
GET https://{endpoint}/v5/iot/{project_id}/bridges
```

响应示例

状态码： 200

OK

```
{
  "bridges" : [ {
    "bridge_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "bridge_name" : "dianabridge",
    "status" : "ONLINE"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListBridgesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListBridgesRequest request = new ListBridgesRequest();
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListBridgesResponse response = client.listBridges(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```


Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListBridgesRequest()
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_bridges(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"
```



```
    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListBridgesRequest{}
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListBridges(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.23.3 删除网桥

功能介绍

应用服务器可调用此接口在物联网平台上删除指定网桥。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/bridges/{bridge_id}

表 1-726 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
bridge_id	是	String	网桥ID。**取值范围**：长度不超过36，只允许字母、数字、_-字符的组合。

请求参数

表 1-727 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

```
DELETE https://{endpoint}/v5/iot/{project_id}/bridges/{bridge_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteBridgeSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteBridgeRequest request = new DeleteBridgeRequest();
        try {
            DeleteBridgeResponse response = client.deleteBridge(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```


Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteBridgeRequest()
        response = client.delete_bridge(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```

```
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteBridgeRequest{}
    response, err := client.DeleteBridge(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.23.4 重置网桥密钥

功能介绍

应用服务器可调用此接口在物联网平台上重置网桥密钥。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/bridges/{bridge_id}/reset-secret

表 1-728 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
bridge_id	是	String	网桥ID。**取值范围**：长度不超过36，只允许字母、数字、_-字符的组合。

请求参数

表 1-729 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-730 请求 Body 参数

参数	是否必选	参数类型	描述
force_disconnect	否	Boolean	是否强制断开网桥的连接，当前仅限长连接。 缺省值：**false**

响应参数

状态码： 200

表 1-731 响应 Body 参数

参数	参数类型	描述
bridge_id	String	网桥ID
secret	String	网桥密钥。

请求示例

```
POST https://{endpoint}/v5/iot/{project_id}/bridges/{bridge_id}/reset-secret

{
  "force_disconnect" : false
}
```

响应示例

状态码： 200

OK

```
{
  "bridge_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "secret" : "3b935a250c50dc2c6d481d048cefdc3c"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ResetBridgeSecretSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
```



```
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ResetBridgeSecretRequest request = new ResetBridgeSecretRequest();
        ResetBridgeSecret body = new ResetBridgeSecret();
        body.withForceDisconnect(false);
        request.withBody(body);
        try {
            ResetBridgeSecretResponse response = client.resetBridgeSecret(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ResetBridgeSecretRequest()
        request.body = ResetBridgeSecret(
            force_disconnect=False
        )
        response = client.reset_bridge_secret(request)
        print(response)
```



```
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ResetBridgeSecretRequest{}
    forceDisconnectResetBridgeSecret:= false
    request.Body = &model.ResetBridgeSecret{
        ForceDisconnect: &forceDisconnectResetBridgeSecret,
    }
    response, err := client.ResetBridgeSecret(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24 设备策略管理

1.4.24.1 创建设备策略

功能介绍

应用服务器可调用此接口在物联网平台创建一个策略，该策略需要绑定到设备和产品下才能生效。

- 一个实例最多能创建50个设备策略。
- 仅**标准版实例、企业版实例**支持该接口调用，基础版不支持。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-policies

表 1-732 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-733 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-734 请求 Body 参数

参数	是否必选	参数类型	描述
policy_name	是	String	**参数说明**：策略名称。 **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。
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，建议携带该参数指定创建的设备归属到哪个资源空间下，否则创建的设备将会归属到**默认资源空间**下。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
statement	是	Array of **Statement** objects	**参数说明**：策略文档。

表 1-735 Statement

参数	是否必选	参数类型	描述
effect	是	String	指定是允许还是拒绝该操作。既有允许（ALLOW）又有拒绝（DENY）的授权语句时，遵循拒绝（DENY）优先的原则。 • ALLOW：允许。 • DENY：拒绝。
actions	是	Array of strings	用于指定策略允许或拒绝的操作。格式为：服务名:资源:操作。当前支持的操作类型如下： • iotda:devices:publish：设备使用MQTT协议发布消息。 • iotda:devices:subscribe：设备使用MQTT协议订阅消息。
resources	是	Array of strings	用于指定允许或拒绝对其执行操作的资源。格式为：资源类型:资源名称。如设备订阅的资源为：topic:/v1/${devices.deviceId}/test/hello。 **取值范围**：资源列表长度最小为1，最大为10，列表中的资源取值范围：仅支持字母，数字，以及/{}$=+#?*:._-组合。

响应参数

状态码： 201

表 1-736 响应 Body 参数

参数	参数类型	描述
app_id	String	**参数说明**：资源空间ID。
policy_id	String	策略ID。
policy_name	String	策略名称。
statement	Array of **Statement** objects	策略文档。
create_time	String	在物联网平台创建策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

参数	参数类型	描述
update_time	String	在物联网平台更新策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-737 Statement

参数	参数类型	描述
effect	String	指定是允许还是拒绝该操作。既有允许（ALLOW）又有拒绝（DENY）的授权语句时，遵循拒绝（DENY）优先的原则。 ● ALLOW：允许。 ● DENY：拒绝。
actions	Array of strings	用于指定策略允许或拒绝的操作。格式为：服务名:资源:操作。当前支持的操作类型如下： ● iotda:devices:publish：设备使用MQTT协议发布消息。 ● iotda:devices:subscribe：设备使用MQTT协议订阅消息。
resources	Array of strings	用于指定允许或拒绝对其执行操作的资源。格式为：资源类型:资源名称。如设备订阅的资源为：topic:/v1/${devices.deviceId}/test/hello。 **取值范围**：资源列表长度最小为1，最大为10，列表中的资源取值范围：仅支持字母，数字，以及/{}$=+#?*:._-组合。

请求示例

- 创建设备策略-允许设备订阅与发布指定topic

```
POST https://{endpoint}/v5/iot/{project_id}/device-policies

{
  "policy_name" : "myPolicyAllow",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "statement" : [ {
    "effect" : "ALLOW",
    "actions" : [ "iotda:devices:publish\niotda:devices:subscribe" ],
    "resources" : [ "topic:/v1/${devices.deviceId}/test/allow" ]
  } ]
}
```

- 创建设备策略-不允许设备订阅与发布指定topic

```
POST https://{endpoint}/v5/iot/{project_id}/device-policies

{
  "policy_name" : "myPolicyDeny",
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "statement" : [ {
    "effect" : "DENY",
```



```
    "actions" : [ "iotda:devices:publish\niotda:devices:subscribe" ],
    "resources" : [ "topic:/v1/${devices.deviceId}/test/deny" ]
  } ]
}
```

响应示例

状态码： 201

Created

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "policy_id" : "5c90fa7d3c4e4405e8525079",
  "policy_name" : "testPolicy",
  "statement" : [ {
    "effect" : "ALLOW",
    "actions" : [ "iotda:devices:publish", "iotda:devices:subscribe" ],
    "resources" : [ "topic:/v1/${devices.deviceId}/test/hello", "topic:/v1/${devices.productId}/test/hello" ]
  } ],
  "create_time" : "20230810T070547Z",
  "update_time" : "20230810T070547Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建设备策略-允许设备订阅与发布指定topic

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);
```

```
        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateDevicePolicyRequest request = new CreateDevicePolicyRequest();
        CreateDevicePolicy body = new CreateDevicePolicy();
        List<String> listStatementResources = new ArrayList<>();
        listStatementResources.add("topic:/v1/${devices.deviceId}/test/allow");
        List<String> listStatementActions = new ArrayList<>();
        listStatementActions.add("iotda:devices:publish
iotda:devices:subscribe");
        List<Statement> listbodyStatement = new ArrayList<>();
        listbodyStatement.add(
            new Statement()
                .withEffect("ALLOW")
                .withActions(listStatementActions)
                .withResources(listStatementResources)
        );
        body.withStatement(listbodyStatement);
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withPolicyName("myPolicyAllow");
        request.withBody(body);
        try {
            CreateDevicePolicyResponse response = client.createDevicePolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建设备策略-不允许设备订阅与发布指定topic

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
```

```
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in
derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateDevicePolicyRequest request = new CreateDevicePolicyRequest();
        CreateDevicePolicy body = new CreateDevicePolicy();
        List<String> listStatementResources = new ArrayList<>();
        listStatementResources.add("topic:/v1/${devices.deviceId}/test/deny");
        List<String> listStatementActions = new ArrayList<>();
        listStatementActions.add("iotda:devices:publish
iotda:devices:subscribe");
        List<Statement> listbodyStatement = new ArrayList<>();
        listbodyStatement.add(
            new Statement()
                .withEffect("DENY")
                .withActions(listStatementActions)
                .withResources(listStatementResources)
        );
        body.withStatement(listbodyStatement);
        body.withAppId("jeQDJQZltU8iKgFFoW060F5SGZka");
        body.withPolicyName("myPolicyDeny");
        request.withBody(body);
        try {
            CreateDevicePolicyResponse response = client.createDevicePolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建设备策略-允许设备订阅与发布指定topic

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
```

```
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateDevicePolicyRequest()
        listResourcesStatement = [
            "topic:/v1/${devices.deviceId}/test/allow"
        ]
        listActionsStatement = [
            "iotda:devices:publish
        iotda:devices:subscribe"
        ]
        listStatementbody = [
            Statement(
                effect="ALLOW",
                actions=listActionsStatement,
                resources=listResourcesStatement
            )
        ]
        request.body = CreateDevicePolicy(
            statement=listStatementbody,
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            policy_name="myPolicyAllow"
        )
        response = client.create_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建设备策略-不允许设备订阅与发布指定topic

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
```

```
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateDevicePolicyRequest()
        listResourcesStatement = [
            "topic:/v1/${devices.deviceId}/test/deny"
        ]
        listActionsStatement = [
            "iotda:devices:publish
        iotda:devices:subscribe"
        ]
        listStatementbody = [
            Statement(
                effect="DENY",
                actions=listActionsStatement,
                resources=listResourcesStatement
            )
        ]
        request.body = CreateDevicePolicy(
            statement=listStatementbody,
            app_id="jeQDJQZltU8iKgFFoW060F5SGZka",
            policy_name="myPolicyDeny"
        )
        response = client.create_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建设备策略-允许设备订阅与发布指定topic

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateDevicePolicyRequest{}
    var listResourcesStatement = []string{
        "topic:/v1/${devices.deviceId}/test/allow",
    }
    var listActionsStatement = []string{
        "iotda:devices:publish
    iotda:devices:subscribe",
    }
    var listStatementbody = []model.Statement{
        {
            Effect: "ALLOW",
            Actions: listActionsStatement,
            Resources: listResourcesStatement,
        },
    }
    appIdCreateDevicePolicy:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    request.Body = &model.CreateDevicePolicy{
        Statement: listStatementbody,
        AppId: &appIdCreateDevicePolicy,
        PolicyName: "myPolicyAllow",
    }
    response, err := client.CreateDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建设备策略-不允许设备订阅与发布指定topic

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
```

```
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateDevicePolicyRequest{}
    var listResourcesStatement = List<String>{
        "topic:/v1/${devices.deviceId}/test/deny",
    }
    var listActionsStatement = List<String>{
        "iotda:devices:publish
    iotda:devices:subscribe",
    }
    var listStatementbody = []model.Statement{
        {
            Effect: "DENY",
            Actions: listActionsStatement,
            Resources: listResourcesStatement,
        },
    }
    appIdCreateDevicePolicy:= "jeQDJQZltU8iKgFFoW060F5SGZka"
    request.Body = &model.CreateDevicePolicy{
        Statement: listStatementbody,
        AppId: &appIdCreateDevicePolicy,
        PolicyName: "myPolicyDeny",
    }
    response, err := client.CreateDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24.2 查询设备策略列表

功能介绍

应用服务器可调用此接口在物联网平台查询策略列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-policies

表 1-738 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-739 Query 参数

参数	是否必选	参数类型	描述
app_id	否	String	**参数说明**：资源空间ID。此参数为非必选参数，存在多资源空间的用户需要使用该接口时，可以携带该参数查询指定资源空间下的设备策略，不携带该参数则会查询该用户下所有设备策略。 **取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。
policy_name	否	String	**参数说明**：策略名称。 **取值范围**：长度不超过256，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-740 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-741 响应 Body 参数

参数	参数类型	描述
policies	Array of **ListDevicePolicyBase** objects	策略信息列表。
page	**Page** object	查询结果的分页信息。

表 1-742 ListDevicePolicyBase

参数	参数类型	描述
app_id	String	**参数说明**：资源空间ID。
policy_id	String	策略ID。
policy_name	String	策略名称。
create_time	String	在物联网平台创建策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

参数	参数类型	描述
update_time	String	在物联网平台更新策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-743 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

查询指定应用下的设备策略列表

```
GET https://{endpoint}/v5/iot/{project_id}/device-policies?app_id={app_id}
```

响应示例

状态码： 200

OK

```
{
  "policies" : [ {
    "policy_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "policy_name" : "myPolicy",
    "create_time" : "20190303T081011Z",
    "update_time" : "20190303T081011Z"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24.3 删除设备策略

功能介绍

应用服务器可调用此接口在物联网平台上删除指定策略，注意：删除策略同时会解绑该策略下所有绑定对象。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/device-policies/{policy_id}

表 1-744 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	策略ID。**取值范围**：仅允许A-F,a-f和数字的组合，长度为24。

请求参数

表 1-745 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除指定id的设备策略

```
DELETE https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteDevicePolicyRequest request = new DeleteDevicePolicyRequest();
        try {
            DeleteDevicePolicyResponse response = client.deleteDevicePolicy(request);
```

```
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteDevicePolicyRequest()
        response = client.delete_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)
```

```
func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteDevicePolicyRequest{}
    response, err := client.DeleteDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24.4 查询设备策略详情

功能介绍

应用服务器可调用此接口在物联网平台查询指定策略ID的详细信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-policies/{policy_id}

表 1-746 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	策略ID

请求参数

表 1-747 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-748 响应 Body 参数

参数	参数类型	描述
app_id	String	**参数说明**：资源空间ID。
policy_id	String	策略ID。
policy_name	String	策略名称。
statement	Array of **Statement** objects	策略文档。
create_time	String	在物联网平台创建策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
update_time	String	在物联网平台更新策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-749 Statement

参数	参数类型	描述
effect	String	指定是允许还是拒绝该操作。既有允许（ALLOW）又有拒绝（DENY）的授权语句时，遵循拒绝（DENY）优先的原则。 ● ALLOW：允许。 ● DENY：拒绝。
actions	Array of strings	用于指定策略允许或拒绝的操作。格式为：服务名:资源:操作。当前支持的操作类型如下： ● iotda:devices:publish：设备使用MQTT协议发布消息。 ● iotda:devices:subscribe：设备使用MQTT协议订阅消息。
resources	Array of strings	用于指定允许或拒绝对其执行操作的资源。格式为：资源类型:资源名称。如设备订阅的资源为：topic:/v1/${devices.deviceId}/test/hello。 **取值范围**：资源列表长度最小为1，最大为10，列表中的资源取值范围：仅支持字母，数字，以及/{}$=+#?*:._-组合。

请求示例

查询指定id的设备策略详情

```
GET https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}
```


响应示例

状态码： 200

OK

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "policy_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "policy_name" : "myPolicy",
  "statement" : [ {
    "effect" : "ALLOW",
    "actions" : [ "iotda:devices:publish", "iotda:devices:subscribe" ],
    "resources" : [ "topic:/v1/${devices.deviceId}/test/hello", "topic:/v1/${devices.productId}/test/hello" ]
  } ],
  "create_time" : "20230810T070547Z",
  "update_time" : "20230810T070547Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDevicePolicyRequest request = new ShowDevicePolicyRequest();
        try {
            ShowDevicePolicyResponse response = client.showDevicePolicy(request);
```

```
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDevicePolicyRequest()
        response = client.show_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)
```

```
func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDevicePolicyRequest{}
    response, err := client.ShowDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24.5 更新设备策略信息

功能介绍

应用服务器可调用此接口在物联网平台更新策略。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/device-policies/{policy_id}

表 1-750 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	策略ID

请求参数

表 1-751 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-752 请求 Body 参数

参数	是否必选	参数类型	描述
policy_name	否	String	**参数说明**：策略名称。 **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。
statement	否	Array of **Statement** objects	**参数说明**：策略文档。

表 1-753 Statement

参数	是否必选	参数类型	描述
effect	是	String	指定是允许还是拒绝该操作。既有允许（ALLOW）又有拒绝（DENY）的授权语句时，遵循拒绝（DENY）优先的原则。 • ALLOW：允许。 • DENY：拒绝。
actions	是	Array of strings	用于指定策略允许或拒绝的操作。格式为：服务名:资源:操作。当前支持的操作类型如下： • iotda:devices:publish：设备使用MQTT协议发布消息。 • iotda:devices:subscribe：设备使用MQTT协议订阅消息。
resources	是	Array of strings	用于指定允许或拒绝对其执行操作的资源。格式为：资源类型:资源名称。如设备订阅的资源为：topic:/v1/${devices.deviceId}/test/hello。 **取值范围**：资源列表长度最小为1，最大为10，列表中的资源取值范围：仅支持字母，数字，以及/{}$=+#?*:._-组合。

响应参数

状态码： 200

表 1-754 响应 Body 参数

参数	参数类型	描述
app_id	String	**参数说明**：资源空间ID。
policy_id	String	策略ID。
policy_name	String	策略名称。
statement	Array of **Statement** objects	策略文档。
create_time	String	在物联网平台创建策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
update_time	String	在物联网平台更新策略的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-755 Statement

参数	参数类型	描述
effect	String	指定是允许还是拒绝该操作。既有允许（ALLOW）又有拒绝（DENY）的授权语句时，遵循拒绝（DENY）优先的原则。 ● ALLOW：允许。 ● DENY：拒绝。
actions	Array of strings	用于指定策略允许或拒绝的操作。格式为：服务名:资源:操作。当前支持的操作类型如下： ● iotda:devices:publish：设备使用MQTT协议发布消息。 ● iotda:devices:subscribe：设备使用MQTT协议订阅消息。
resources	Array of strings	用于指定允许或拒绝对其执行操作的资源。格式为：资源类型:资源名称。如设备订阅的资源为：topic:/v1/${devices.deviceId}/test/hello。 **取值范围**：资源列表长度最小为1，最大为10，列表中的资源取值范围：仅支持字母，数字，以及/{}$=+#?*:._-组合。

请求示例

更新设备策略

```
PUT https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}
```

```
{
  "policy_name" : "testPolicy",
  "statement" : [ {
    "effect" : "ALLOW",
    "actions" : [ "iotda:devices:publish", "iotda:devices:subscribe" ],
    "resources" : [ "topic:/v1/${devices.deviceId}/test/hello", "topic:/v1/${devices.productId}/test/hello" ]
  } ]
}
```

响应示例

状态码： 200

OK

```
{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "policy_id" : "5c90fa7d3c4e4405e8525079",
  "policy_name" : "testPolicy",
  "statement" : [ {
    "effect" : "ALLOW",
    "actions" : [ "iotda:devices:publish", "iotda:devices:subscribe" ],
    "resources" : [ "topic:/v1/${devices.deviceId}/test/hello", "topic:/v1/${devices.productId}/test/hello" ]
  } ],
  "create_time" : "20230810T070547Z",
  "update_time" : "20230810T070547Z"
}
```

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24.6 绑定设备策略

功能介绍

应用服务器可调用此接口在物联网平台上为批量设备绑定目标策略，目前支持绑定目标类型为：设备、产品，当目标类型为产品时，该产品下所有设备都会生效。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-policies/{policy_id}/bind

表 1-756 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	策略ID

请求参数

表 1-757 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-758 请求 Body 参数

参数	是否必选	参数类型	描述
target_type	是	String	**参数说明**：策略绑定的目标类型。 **取值范围**：device|product|app，device表示设备，product表示产品，app表示整个资源空间。
target_ids	是	Array of strings	策略绑定的目标ID列表

响应参数

状态码： 200

表 1-759 响应 Body 参数

参数	参数类型	描述
policy_id	String	策略ID。
target_type	String	**参数说明**：策略的目标类型。 **取值范围**：device|product|app，device表示设备，product表示产品，app表示整个资源空间。
success_targets	Array of strings	成功的目标id列表。
failure_targets	Array of **DevicePolicyBindOrUnbindFailureDetail** objects	失败的目标id列表

表 1-760 DevicePolicyBindOrUnbindFailureDetail

参数	参数类型	描述
target_id	String	失败的目标id。
error_code	String	错误码。
error_msg	String	错误详情。

请求示例

- 为设备绑定策略

```
POST https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}/bind

{
  "target_type" : "device",
  "target_ids" : [ "64a7ba7ef9cb063d27e16b97_123456", "64a7ba7ef9cb063d27e16b97_123457" ]
}
```

- 为产品绑定策略

```
POST https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}/bind

{
  "target_type" : "product",
  "target_ids" : [ "64a7ba7ef9cb063d27e16b97" ]
}
```

响应示例

状态码： 200

OK


```
{
  "policy_id" : "5c90fa7d3c4e4405e8525079",
  "target_type" : "device",
  "success_targets" : [ "64a7ba7ef9cb063d27e16b97_123456", "64a7ba7ef9cb063d27e16b97_123457" ],
  "failure_targets" : [ {
    "target_id" : "64a7ba7ef9cb063d27e16b97_123458",
    "error_code" : "IOTDA.015204",
    "error_msg" : "Operation not allowed. The number of policies bound to the target reaches the upper limit (5)."
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 为设备绑定策略

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class BindDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        BindDevicePolicyRequest request = new BindDevicePolicyRequest();
        BindDevicePolicy body = new BindDevicePolicy();
        List<String> listbodyTargetIds = new ArrayList<>();
        listbodyTargetIds.add("64a7ba7ef9cb063d27e16b97_123456");
        listbodyTargetIds.add("64a7ba7ef9cb063d27e16b97_123457");
        body.withTargetIds(listbodyTargetIds);
```

```
        body.withTargetType("device");
        request.withBody(body);
        try {
            BindDevicePolicyResponse response = client.bindDevicePolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 为产品绑定策略

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class BindDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        BindDevicePolicyRequest request = new BindDevicePolicyRequest();
        BindDevicePolicy body = new BindDevicePolicy();
        List<String> listbodyTargetIds = new ArrayList<>();
        listbodyTargetIds.add("64a7ba7ef9cb063d27e16b97");
        body.withTargetIds(listbodyTargetIds);
        body.withTargetType("product");
        request.withBody(body);
        try {
```

```
            BindDevicePolicyResponse response = client.bindDevicePolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 为设备绑定策略

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = BindDevicePolicyRequest()
        listTargetIdsbody = [
            "64a7ba7ef9cb063d27e16b97_123456",
            "64a7ba7ef9cb063d27e16b97_123457"
        ]
        request.body = BindDevicePolicy(
            target_ids=listTargetIdsbody,
            target_type="device"
        )
        response = client.bind_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```


- 为产品绑定策略

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = BindDevicePolicyRequest()
        listTargetIdsbody = [
            "64a7ba7ef9cb063d27e16b97"
        ]
        request.body = BindDevicePolicy(
            target_ids=listTargetIdsbody,
            target_type="product"
        )
        response = client.bind_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 为设备绑定策略

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
```

```
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.BindDevicePolicyRequest{}
    var listTargetIdsbody = []string{
        "64a7ba7ef9cb063d27e16b97_123456",
        "64a7ba7ef9cb063d27e16b97_123457",
    }
    request.Body = &model.BindDevicePolicy{
        TargetIds: listTargetIdsbody,
        TargetType: "device",
    }
    response, err := client.BindDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 为产品绑定策略

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()
```



```
    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.BindDevicePolicyRequest{}
    var listTargetIdsbody = []string{
        "64a7ba7ef9cb063d27e16b97",
    }
    request.Body = &model.BindDevicePolicy{
        TargetIds: listTargetIdsbody,
        TargetType: "product",
    }
    response, err := client.BindDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24.7 解绑设备策略

功能介绍

应用服务器可调用此接口在物联网平台上解除指定策略下绑定的目标对象。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-policies/{policy_id}/unbind

表 1-761 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	策略ID

请求参数

表 1-762 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-763 请求 Body 参数

参数	是否必选	参数类型	描述
target_type	是	String	**参数说明**：解除策略绑定的目标类型。 **取值范围**：device|product|app，device表示设备，product表示产品，app表示整个资源空间。
target_ids	是	Array of strings	策略解绑的目标ID列表

响应参数

状态码： 200

表 1-764 响应 Body 参数

参数	参数类型	描述
policy_id	String	策略ID。
target_type	String	**参数说明**：策略的目标类型。 **取值范围**：device|product|app，device表示设备，product表示产品，app表示整个资源空间。
success_targets	Array of strings	成功的目标id列表。
failure_targets	Array of **DevicePolicyBindOrUnbindFailureDetail** objects	失败的目标id列表

表 1-765 DevicePolicyBindOrUnbindFailureDetail

参数	参数类型	描述
target_id	String	失败的目标id。
error_code	String	错误码。
error_msg	String	错误详情。

请求示例

- 解绑设备下的策略

```
POST https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}/unbind

{
  "target_type" : "device",
  "target_ids" : [ "64a7ba7ef9cb063d27e16b97_123456" ]
}
```

- 解绑产品下的策略

```
POST https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}/unbind

{
  "target_type" : "product",
  "target_ids" : [ "64a7ba7ef9cb063d27e16b97" ]
}
```

响应示例

状态码： 200

OK

```
{
  "policy_id" : "5c90fa7d3c4e4405e8525079",
  "target_type" : "device",
  "success_targets" : [ "64a7ba7ef9cb063d27e16b97_123456", "64a7ba7ef9cb063d27e16b97_123457" ],
  "failure_targets" : [ {
    "target_id" : "64a7ba7ef9cb063d27e16b97_123458",
    "error_code" : "IOTDA.015207",
    "error_msg" : "Invalid input. The target does not bind the device-policy."
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 解绑设备下的策略

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UnbindDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UnbindDevicePolicyRequest request = new UnbindDevicePolicyRequest();
        UnBindDevicePolicy body = new UnBindDevicePolicy();
        List<String> listbodyTargetIds = new ArrayList<>();
        listbodyTargetIds.add("64a7ba7ef9cb063d27e16b97_123456");
        body.withTargetIds(listbodyTargetIds);
        body.withTargetType("device");
        request.withBody(body);
```

```
        try {
            UnbindDevicePolicyResponse response = client.unbindDevicePolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 解绑产品下的策略

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UnbindDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UnbindDevicePolicyRequest request = new UnbindDevicePolicyRequest();
        UnBindDevicePolicy body = new UnBindDevicePolicy();
        List<String> listbodyTargetIds = new ArrayList<>();
        listbodyTargetIds.add("64a7ba7ef9cb063d27e16b97");
        body.withTargetIds(listbodyTargetIds);
        body.withTargetType("product");
        request.withBody(body);
        try {
            UnbindDevicePolicyResponse response = client.unbindDevicePolicy(request);
            System.out.println(response.toString());
```

```
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 解绑设备下的策略

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UnbindDevicePolicyRequest()
        listTargetIdsbody = [
            "64a7ba7ef9cb063d27e16b97_123456"
        ]
        request.body = UnBindDevicePolicy(
            target_ids=listTargetIdsbody,
            target_type="device"
        )
        response = client.unbind_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 解绑产品下的策略

```
# coding: utf-8
```

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UnbindDevicePolicyRequest()
        listTargetIdsbody = [
            "64a7ba7ef9cb063d27e16b97"
        ]
        request.body = UnBindDevicePolicy(
            target_ids=listTargetIdsbody,
            target_type="product"
        )
        response = client.unbind_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 解绑设备下的策略

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
```

```
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UnbindDevicePolicyRequest{}
    var listTargetIdsbody = []string{
        "64a7ba7ef9cb063d27e16b97_123456",
    }
    request.Body = &model.UnBindDevicePolicy{
        TargetIds: listTargetIdsbody,
        TargetType: "device",
    }
    response, err := client.UnbindDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 解绑产品下的策略

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
```

```
        WithCredential(auth).
        Build())

    request := &model.UnbindDevicePolicyRequest{}
    var listTargetIdsbody = []string{
        "64a7ba7ef9cb063d27e16b97",
    }
    request.Body = &model.UnBindDevicePolicy{
        TargetIds: listTargetIdsbody,
        TargetType: "product",
    }
    response, err := client.UnbindDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.24.8 查询设备策略绑定的目标列表

功能介绍

应用服务器可调用此接口在物联网平台上查询指定策略ID下绑定的目标列表。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-policies/{policy_id}/list-targets

表 1-766 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
policy_id	是	String	策略ID

请求参数

表 1-767 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-768 请求 Body 参数

参数	是否必选	参数类型	描述
target_type	否	String	**参数说明**：策略绑定的目标类型。 **取值范围**：device|product|app，device表示设备，product表示产品，app表示整个资源空间。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

响应参数

状态码： 200

表 1-769 响应 Body 参数

参数	参数类型	描述
targets	Array of **PolicyTargetBase** objects	策略绑定信息列表。
page	**Page** object	查询结果的分页信息。

表 1-770 PolicyTargetBase

参数	参数类型	描述
target_type	String	**参数说明**：策略绑定的目标类型。 **取值范围**：device|product|app，device表示设备，product表示产品，app表示整个资源空间。
target_id	String	策略绑定的目标ID

表 1-771 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

查询指定策略id下绑定的目标列表

```
POST https://{endpoint}/v5/iot/{project_id}/device-policies/{policy_id}/list-targets

{
  "target_type" : "device"
}
```

响应示例

状态码： 200

OK

```
{
  "targets" : [ {
    "target_type" : "device",
    "target_id" : "64a7ba7ef9cb063d27e16b97_123456"
  } ],
  "page" : {
    "marker" : "5c90fa7d3c4e4405e8525079",
    "count" : 1
```

```
 }
}
```

SDK 代码示例

SDK代码示例如下。

Java

查询指定策略id下绑定的目标列表

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowTargetsInDevicePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowTargetsInDevicePolicyRequest request = new ShowTargetsInDevicePolicyRequest();
        ShowTargetsInDevicePolicyRequestBody body = new ShowTargetsInDevicePolicyRequestBody();
        body.withTargetType("device");
        request.withBody(body);
        try {
            ShowTargetsInDevicePolicyResponse response = client.showTargetsInDevicePolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
```



```
        }
    }
}
```

Python

查询指定策略id下绑定的目标列表

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowTargetsInDevicePolicyRequest()
        request.body = ShowTargetsInDevicePolicyRequestBody(
            target_type="device"
        )
        response = client.show_targets_in_device_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

查询指定策略id下绑定的目标列表

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
```

```
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowTargetsInDevicePolicyRequest{}
    targetTypeShowTargetsInDevicePolicyRequestBody:= "device"
    request.Body = &model.ShowTargetsInDevicePolicyRequestBody{
        TargetType: &targetTypeShowTargetsInDevicePolicyRequestBody,
    }
    response, err := client.ShowTargetsInDevicePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.25 预调配模板管理

1.4.25.1 创建预调配模板

功能介绍

应用服务器可调用此接口在物联网平台创建一个预调配模板。用户的设备未在平台注册时，可以通过预调配模板在设备首次接入物联网平台时将设备信息自动注册到物联网平台。

- 该预调配模板至少需要绑定到一个设备CA证书下才能生效。
- 一个实例最多可有10个预调配模板。
- 仅标准版实例、企业版实例支持该接口调用，基础版不支持。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/provisioning-templates

表 1-772 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-773 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。

参数	是否必选	参数类型	描述
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-774 请求 Body 参数

参数	是否必选	参数类型	描述
template_name	是	String	**参数说明**：预调配模板名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、下划线（_）、连接符（-）的组合。
description	否	String	**参数说明**：预调配模板的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
template_body	是	**ProvisioningTemplateBody** object	**参数说明**：预调配模板详细内容，json格式。

表 1-775 ProvisioningTemplateBody

参数	是否必选	参数类型	描述
parameters	是	Object	**参数说明**：预调配模板参数，，配置格式为{"parameter":{"type":"String"}}，其中parameter目前支持从预调配设备的证书的使用者字段提取内容，证书必须包含模板中定义的所有参数值，华为云IoT平台定义了可在预调配模板中声明和引用的如下参数: ● iotda::certificate::country (国家/地区,C) ● iotda::certificate::organization (组织,O) ● iotda::certificate::organizational_unit (组织单位,OU) ● iotda::certificate::distinguished_name_qualifier (可辨别名称限定符,dnQualifier) ● iotda::certificate::state_name (省市,ST) ● iotda::certificate::common_name (公用名,CN) ● iotda::certificate::serial_number (序列号,serialNumber) type描述parameter的类型，目前仅支持string。 配置样例： '{"iotda::certificate::country":{"type":"String"}, "iotda::certificate::organization":{"type":"String"}, "iotda::certificate::organizational_unit":{"type":"String"}, "iotda::certificate::distinguished_name_qualifier":{"type":"String"}, "iotda::certificate::state_name":{"type":"String"}, "iotda::certificate::common_name":{"type":"String"}, "iotda::certificate::serial_number":{"type":"String"}}'

参数	是否必选	参数类型	描述
resources	是	**TemplateResource** object	预调配模板设备资源结构体。

表 1-776 TemplateResource

参数	是否必选	参数类型	描述
device	是	**DeviceResource** object	预调配模板设备资源详情结构体。
policy	否	**PolicyResource** object	预调配模板设备策略资源详情结构体。

表 1-777 DeviceResource

参数	是否必选	参数类型	描述
device_name	否	**ParameterRef** object	设备名称
node_id	是	**ParameterRef** object	设备标识码
product_id	是	Object	**参数说明**：设备所属的产品id，可以是一个明确的静态字符串id，也可以是动态的模板参数引用 • 明确的静态字符串："642bf260f2f9030e44210d8d"。**取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。" • 参数引用: {"ref" : "iotda::certificate::country"}
tags	否	Array of **TagRef** objects	**参数说明**：设备绑定的标签列表

表 1-778 ParameterRef

参数	是否必选	参数类型	描述
ref	是	String	参数引用名称

表 1-779 TagRef

参数	是否必选	参数类型	描述
tag_key	否	Object	**参数说明**：标签键名称，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagKey"。**取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合 ● 参数引用: {"ref" : "iotda::certificate::country"}
tag_value	否	Object	**参数说明**：标签值，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagValue"。**取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。 ● 参数引用: {"ref" : "iotda::certificate::country"}

表 1-780 PolicyResource

参数	是否必选	参数类型	描述
policy_ids	否	Array of strings	**参数说明**：设备需要绑定的策略id列表

响应参数

状态码： 201

表 1-781 响应 Body 参数

参数	参数类型	描述
template_id	String	**参数说明**：预调配模板ID。
template_name	String	**参数说明**：预调配模板名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、下划线（_）、连接符（-）的组合。

参数	参数类型	描述
description	String	**参数说明**：预调配模板的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
template_body	**ProvisioningTemplateBody** object	**参数说明**：预调配模板详细内容，json格式。
create_time	String	在物联网平台创建预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
update_time	String	在物联网平台更新预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-782 ProvisioningTemplateBody

参数	参数类型	描述
parameters	Object	**参数说明**：预调配模板参数，，配置格式为{"parameter":{"type":"String"}}，其中parameter目前支持从预调配设备的证书的使用者字段提取内容，证书必须包含模板中定义的所有参数值，华为云IoT平台定义了可在预调配模板中声明和引用的如下参数: • iotda::certificate::country (国家/地区,C) • iotda::certificate::organization (组织,O) • iotda::certificate::organizational_unit (组织单位,OU) • iotda::certificate::distinguished_name_qualifier (可辨别名称限定符,dnQualifier) • iotda::certificate::state_name (省市,ST) • iotda::certificate::common_name (公用名,CN) • iotda::certificate::serial_number (序列号,serialNumber) type描述parameter的类型，目前仅支持string。 配置样例: '{"iotda::certificate::country":{"type":"String"}, "iotda::certificate::organization":{"type":"String"}, "iotda::certificate::organizational_unit":{"type":"String"}, "iotda::certificate::distinguished_name_qualifier":{"type":"String"}, "iotda::certificate::state_name":{"type":"String"}, "iotda::certificate::common_name":{"type":"String"}, "iotda::certificate::serial_number":{"type":"String"}}'
resources	**TemplateResource** object	预调配模板设备资源结构体。

表 1-783 TemplateResource

参数	参数类型	描述
device	**DeviceResource** object	预调配模板设备资源详情结构体。

参数	参数类型	描述
policy	**PolicyResource** object	预调配模板设备策略资源详情结构体。

表 1-784 DeviceResource

参数	参数类型	描述
device_name	**ParameterRef** object	设备名称
node_id	**ParameterRef** object	设备标识码
product_id	Object	**参数说明**：设备所属的产品id，可以是一个明确的静态字符串id，也可以是动态的模板参数引用 ● 明确的静态字符串："642bf260f2f9030e44210d8d"。**取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。" ● 参数引用: {"ref" : "iotda::certificate::country"}
tags	Array of **TagRef** objects	**参数说明**：设备绑定的标签列表

表 1-785 ParameterRef

参数	参数类型	描述
ref	String	参数引用名称

表 1-786 TagRef

参数	参数类型	描述
tag_key	Object	**参数说明**：标签键名称，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagKey"。**取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合 ● 参数引用: {"ref" : "iotda::certificate::country"}

参数	参数类型	描述
tag_value	Object	**参数说明**：标签值，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagValue"。**取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。 ● 参数引用: {"ref" : "iotda::certificate::country"}

表 1-787 PolicyResource

参数	参数类型	描述
policy_ids	Array of strings	**参数说明**：设备需要绑定的策略id列表

请求示例

- 创建预调配模板-参数引用

```
POST https://{endpoint}/v5/iot/{project_id}/provisioning-templates

{
  "template_name" : "myTemplate",
  "description" : "myTemplate",
  "template_body" : {
    "parameters" : {
      "iotda::certificate::country" : {
        "type" : "String"
      },
      "iotda::certificate::organization" : {
        "type" : "String"
      },
      "iotda::certificate::organizational_unit" : {
        "type" : "String"
      },
      "iotda::certificate::distinguished_name_qualifier" : {
        "type" : "String"
      },
      "iotda::certificate::state_name" : {
        "type" : "String"
      },
      "iotda::certificate::common_name" : {
        "type" : "String"
      },
      "iotda::certificate::serial_number" : {
        "type" : "String"
      }
    },
    "resources" : {
      "device" : {
        "device_name" : {
          "ref" : "iotda::certificate::organization"
        },
        "node_id" : {
          "ref" : "iotda::certificate::common_name"
        },
        "product_id" : {
```



```
        "ref" : "iotda::certificate::organization"
      },
      "tags" : [ {
        "tag_key" : {
          "ref" : "iotda::certificate::organization"
        },
        "tag_value" : {
          "ref" : "iotda::certificate::organizational_unit"
        }
      } ]
    },
    "policy" : {
      "policy_ids" : [ "5c90fa7d3c4e4405e8525079" ]
    }
  }
 }
}
```

- 创建预调配模板-部分参数自定义

```
POST https://{endpoint}/v5/iot/{project_id}/provisioning-templates

{
 "template_name" : "myTemplate2",
 "description" : "myTemplate2",
 "template_body" : {
  "parameters" : {
   "iotda::certificate::country" : {
    "type" : "String"
   },
   "iotda::certificate::organization" : {
    "type" : "String"
   },
   "iotda::certificate::organizational_unit" : {
    "type" : "String"
   },
   "iotda::certificate::distinguished_name_qualifier" : {
    "type" : "String"
   },
   "iotda::certificate::state_name" : {
    "type" : "String"
   },
   "iotda::certificate::common_name" : {
    "type" : "String"
   },
   "iotda::certificate::serial_number" : {
    "type" : "String"
   }
  },
  "resources" : {
   "device" : {
    "device_name" : {
     "ref" : "iotda::certificate::organization"
    },
    "node_id" : {
     "ref" : "iotda::certificate::common_name"
    },
    "product_id" : "642bf260f2f9030e44210d8d",
    "tags" : [ {
     "tag_key" : "myTagKey",
     "tag_value" : "myTagValue"
    } ]
   },
   "policy" : {
    "policy_ids" : [ "5c90fa7d3c4e4405e8525079" ]
   }
  }
 }
}
```


响应示例

状态码： 201

Created

```
{
  "template_id" : "5c90fa7d3c4e4405e8525079",
  "template_name" : "myTemplate",
  "description" : "myTemplate",
  "template_body" : {
    "parameters" : {
      "iotda::certificate::country" : {
        "type" : "String"
      },
      "iotda::certificate::organization" : {
        "type" : "String"
      },
      "iotda::certificate::organizational_unit" : {
        "type" : "String"
      },
      "iotda::certificate::distinguished_name_qualifier" : {
        "type" : "String"
      },
      "iotda::certificate::state_name" : {
        "type" : "String"
      },
      "iotda::certificate::common_name" : {
        "type" : "String"
      },
      "iotda::certificate::serial_number" : {
        "type" : "String"
      }
    },
    "resources" : {
      "device" : {
        "device_name" : {
          "ref" : "iotda::certificate::organization"
        },
        "node_id" : {
          "ref" : "iotda::certificate::common_name"
        },
        "product_id" : {
          "ref" : "iotda::certificate::organization"
        }
      },
      "policy" : {
        "policy_ids" : [ "5c90fa7d3c4e4405e8525079" ]
      }
    }
  },
  "create_time" : "20230810T070547Z",
  "update_time" : "20230810T070547Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建预调配模板-参数引用

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
```

```
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateProvisioningTemplateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateProvisioningTemplateRequest request = new CreateProvisioningTemplateRequest();
        CreateProvisioningTemplate body = new CreateProvisioningTemplate();
        List<String> listPolicyPolicyIds = new ArrayList<>();
        listPolicyPolicyIds.add("5c90fa7d3c4e4405e8525079");
        PolicyResource policyResources = new PolicyResource();
        policyResources.withPolicyIds(listPolicyPolicyIds);
        List<TagRef> listDeviceTags = new ArrayList<>();
        listDeviceTags.add(
            new TagRef()
                .withTagKey("{\"ref\":\"iotda::certificate::organization\"}")
                .withTagValue("{\"ref\":\"iotda::certificate::organizational_unit\"}")
        );
        ParameterRef nodeIdDevice = new ParameterRef();
        nodeIdDevice.withRef("iotda::certificate::common_name");
        ParameterRef deviceNameDevice = new ParameterRef();
        deviceNameDevice.withRef("iotda::certificate::organization");
        DeviceResource deviceResources = new DeviceResource();
        deviceResources.withDeviceName(deviceNameDevice)
            .withNodeId(nodeIdDevice)
            .withProductId("{\"ref\":\"iotda::certificate::organization\"}")
            .withTags(listDeviceTags);
        TemplateResource resourcesTemplateBody = new TemplateResource();
        resourcesTemplateBody.withDevice(deviceResources)
            .withPolicy(policyResources);
        ProvisioningTemplateBody templateBodybody = new ProvisioningTemplateBody();
        templateBodybody.withParameters("{\"iotda::certificate::serial_number\":{\"type\":\"String\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}")
                .withResources(resourcesTemplateBody);
        body.withTemplateBody(templateBodybody);
```



```
        body.withDescription("myTemplate");
        body.withTemplateName("myTemplate");
        request.withBody(body);
        try {
            CreateProvisioningTemplateResponse response = client.createProvisioningTemplate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建预调配模板-部分参数自定义

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateProvisioningTemplateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateProvisioningTemplateRequest request = new CreateProvisioningTemplateRequest();
        CreateProvisioningTemplate body = new CreateProvisioningTemplate();
        List<String> listPolicyPolicyIds = new ArrayList<>();
        listPolicyPolicyIds.add("5c90fa7d3c4e4405e8525079");
        PolicyResource policyResources = new PolicyResource();
        policyResources.withPolicyIds(listPolicyPolicyIds);
        List<TagRef> listDeviceTags = new ArrayList<>();
```

```
        listDeviceTags.add(
            new TagRef()
                .withTagKey("myTagKey")
                .withTagValue("myTagValue")
        );
        ParameterRef nodeIdDevice = new ParameterRef();
        nodeIdDevice.withRef("iotda::certificate::common_name");
        ParameterRef deviceNameDevice = new ParameterRef();
        deviceNameDevice.withRef("iotda::certificate::organization");
        DeviceResource deviceResources = new DeviceResource();
        deviceResources.withDeviceName(deviceNameDevice)
            .withNodeId(nodeIdDevice)
            .withProductId("642bf260f2f9030e44210d8d")
            .withTags(listDeviceTags);
        TemplateResource resourcesTemplateBody = new TemplateResource();
        resourcesTemplateBody.withDevice(deviceResources)
            .withPolicy(policyResources);
        ProvisioningTemplateBody templateBodybody = new ProvisioningTemplateBody();
        templateBodybody.withParameters("{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}")
            .withResources(resourcesTemplateBody);
        body.withTemplateBody(templateBodybody);
        body.withDescription("myTemplate2");
        body.withTemplateName("myTemplate2");
        request.withBody(body);
        try {
            CreateProvisioningTemplateResponse response = client.createProvisioningTemplate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 创建预调配模板-参数引用

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";
```

```
    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateProvisioningTemplateRequest()
        listPolicyIdsPolicy = [
            "5c90fa7d3c4e4405e8525079"
        ]
        policyResources = PolicyResource(
            policy_ids=listPolicyIdsPolicy
        )
        listTagsDevice = [
            TagRef(
                tag_key="{\"ref\":\"iotda::certificate::organization\"}",
                tag_value="{\"ref\":\"iotda::certificate::organizational_unit\"}"
            )
        ]
        nodeIdDevice = ParameterRef(
            ref="iotda::certificate::common_name"
        )
        deviceNameDevice = ParameterRef(
            ref="iotda::certificate::organization"
        )
        deviceResources = DeviceResource(
            device_name=deviceNameDevice,
            node_id=nodeIdDevice,
            product_id="{\"ref\":\"iotda::certificate::organization\"}",
            tags=listTagsDevice
        )
        resourcesTemplateBody = TemplateResource(
            device=deviceResources,
            policy=policyResources
        )
        templateBodybody = ProvisioningTemplateBody(
            parameters="{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}",
            resources=resourcesTemplateBody
        )
        request.body = CreateProvisioningTemplate(
            template_body=templateBodybody,
            description="myTemplate",
            template_name="myTemplate"
        )
        response = client.create_provisioning_template(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建预调配模板-部分参数自定义

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *
```

```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateProvisioningTemplateRequest()
        listPolicyIdsPolicy = [
            "5c90fa7d3c4e4405e8525079"
        ]
        policyResources = PolicyResource(
            policy_ids=listPolicyIdsPolicy
        )
        listTagsDevice = [
            TagRef(
                tag_key="myTagKey",
                tag_value="myTagValue"
            )
        ]
        nodeIdDevice = ParameterRef(
            ref="iotda::certificate::common_name"
        )
        deviceNameDevice = ParameterRef(
            ref="iotda::certificate::organization"
        )
        deviceResources = DeviceResource(
            device_name=deviceNameDevice,
            node_id=nodeIdDevice,
            product_id="642bf260f2f9030e44210d8d",
            tags=listTagsDevice
        )
        resourcesTemplateBody = TemplateResource(
            device=deviceResources,
            policy=policyResources
        )
        templateBodybody = ProvisioningTemplateBody(
            parameters="{\"iotda::certificate::serial_number\":{\"type\":\"String\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}",
            resources=resourcesTemplateBody
        )
        request.body = CreateProvisioningTemplate(
            template_body=templateBodybody,
            description="myTemplate2",
            template_name="myTemplate2"
        )
        response = client.create_provisioning_template(request)
        print(response)
    except exceptions.ClientRequestException as e:
```



```
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建预调配模板-参数引用

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.CreateProvisioningTemplateRequest{}
    var listPolicyIdsPolicy = []string{
        "5c90fa7d3c4e4405e8525079",
    }
    policyResources := &model.PolicyResource{
        PolicyIds: &listPolicyIdsPolicy,
    }
    tagKeyTags:= "{\"ref\":\"iotda::certificate::organization\"}"
    var tagKeyTagsInterface interface{} = tagKeyTags
    tagValueTags:= "{\"ref\":\"iotda::certificate::organizational_unit\"}"
    var tagValueTagsInterface interface{} = tagValueTags
    var listTagsDevice = []model.TagRef{
        {
            TagKey: &tagKeyTagsInterface,
            TagValue: &tagValueTagsInterface,
        },
    }
    nodeIdDevice := &model.ParameterRef{
        Ref: "iotda::certificate::common_name",
    }
    deviceNameDevice := &model.ParameterRef{
        Ref: "iotda::certificate::organization",
    }
```



```
    var productIdDevice interface{} = "{\"ref\":\"iotda::certificate::organization\"}"
    deviceResources := &model.DeviceResource{
        DeviceName: deviceNameDevice,
        NodeId: nodeIdDevice,
        ProductId: &productIdDevice,
        Tags: &listTagsDevice,
    }
    resourcesTemplateBody := &model.TemplateResource{
        Device: deviceResources,
        Policy: policyResources,
    }
    var parametersTemplateBody interface{} = "{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}"
    templateBodybody := &model.ProvisioningTemplateBody{
        Parameters: &parametersTemplateBody,
        Resources: resourcesTemplateBody,
    }
    descriptionCreateProvisioningTemplate:= "myTemplate"
    request.Body = &model.CreateProvisioningTemplate{
        TemplateBody: templateBodybody,
        Description: &descriptionCreateProvisioningTemplate,
        TemplateName: "myTemplate",
    }
    response, err := client.CreateProvisioningTemplate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建预调配模板-部分参数自定义

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
```

```
        WithCredential(auth).
        Build())

    request := &model.CreateProvisioningTemplateRequest{}
    var listPolicyIdsPolicy = []string{
        "5c90fa7d3c4e4405e8525079",
    }
    policyResources := &model.PolicyResource{
        PolicyIds: &listPolicyIdsPolicy,
    }
    tagKeyTags:= "myTagKey"
    var tagKeyTagsInterface interface{} = tagKeyTags
    tagValueTags:= "myTagValue"
    var tagValueTagsInterface interface{} = tagValueTags
    var listTagsDevice = []model.TagRef{
        {
            TagKey: &tagKeyTagsInterface,
            TagValue: &tagValueTagsInterface,
        },
    }
    nodeIdDevice := &model.ParameterRef{
        Ref: "iotda::certificate::common_name",
    }
    deviceNameDevice := &model.ParameterRef{
        Ref: "iotda::certificate::organization",
    }
    var productIdDevice interface{} = "642bf260f2f9030e44210d8d"
    deviceResources := &model.DeviceResource{
        DeviceName: deviceNameDevice,
        NodeId: nodeIdDevice,
        ProductId: &productIdDevice,
        Tags: &listTagsDevice,
    }
    resourcesTemplateBody := &model.TemplateResource{
        Device: deviceResources,
        Policy: policyResources,
    }
    var parametersTemplateBody interface{} = "{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}"
    templateBodybody := &model.ProvisioningTemplateBody{
        Parameters: &parametersTemplateBody,
        Resources: resourcesTemplateBody,
    }
    descriptionCreateProvisioningTemplate:= "myTemplate2"
    request.Body = &model.CreateProvisioningTemplate{
        TemplateBody: templateBodybody,
        Description: &descriptionCreateProvisioningTemplate,
        TemplateName: "myTemplate2",
    }
    response, err := client.CreateProvisioningTemplate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
500	Internal Server Error

错误码

请参见**错误码**。

1.4.25.2 查询预调配模板列表

功能介绍

应用服务器可调用此接口在物联网平台查询预调配模板列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/provisioning-templates

表 1-788 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-789 Query 参数

参数	是否必选	参数类型	描述
template_name	否	String	**参数说明**：预调配模板名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、下划线（_）、连接符（-）的组合。

参数	是否必选	参数类型	描述
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-790 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-791 响应 Body 参数

参数	参数类型	描述
templates	Array of **ProvisioningTemplateSimple** objects	**参数说明**：预调配模板详情。
page	**Page** object	查询结果的分页信息。

表 1-792 ProvisioningTemplateSimple

参数	参数类型	描述
template_id	String	**参数说明**：预调配模板ID。
template_name	String	**参数说明**：预调配模板名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、下划线（_）、连接符（-）的组合。

参数	参数类型	描述
description	String	**参数说明**：预调配模板的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
create_time	String	在物联网平台创建预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
update_time	String	在物联网平台更新预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-793 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

查询租户下的预调配模板列表

```
GET https://{endpoint}/v5/iot/{project_id}/provisioning-templates?template_name={template_name}
```

响应示例

状态码： 200

OK

```
{
  "templates" : [ {
    "template_id" : "5c90fa7d3c4e4405e8525079",
    "template_name" : "myTemplate",
    "description" : "myTemplate",
    "create_time" : "20230810T070547Z",
    "update_time" : "20230810T070547Z"
  } ],
  "page" : {
    "count" : 1,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListProvisioningTemplatesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListProvisioningTemplatesRequest request = new ListProvisioningTemplatesRequest();
        request.withTemplateName("<template_name>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListProvisioningTemplatesResponse response = client.listProvisioningTemplates(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
```

```
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListProvisioningTemplatesRequest()
        request.template_name = "<template_name>"
        request.limit = <limit>
        request.marker = "<marker>"
        request.offset = <offset>
        response = client.list_provisioning_templates(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
```

```
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListProvisioningTemplatesRequest{}
    templateNameRequest:= "<template_name>"
    request.TemplateName = &templateNameRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListProvisioningTemplates(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.25.3 删除预调配模板

功能介绍

应用服务器可调用此接口在物联网平台上删除指定预调配模板。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/provisioning-templates/{template_id}

表 1-794 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
template_id	是	String	预调配模板ID

请求参数

表 1-795 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除指定id的预调配模板

```
DELETE https://{endpoint}/v5/iot/{project_id}/provisioning-templates/{template_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class DeleteProvisioningTemplateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteProvisioningTemplateRequest request = new DeleteProvisioningTemplateRequest();
        try {
            DeleteProvisioningTemplateResponse response = client.deleteProvisioningTemplate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *
```

```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteProvisioningTemplateRequest()
        response = client.delete_provisioning_template(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
```

```
        Build())

    request := &model.DeleteProvisioningTemplateRequest{}
    response, err := client.DeleteProvisioningTemplate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.25.4 查询预调配模板详情

功能介绍

应用服务器可调用此接口在物联网平台查询指定预调配模板ID的详细信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/provisioning-templates/{template_id}

表 1-796 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
template_id	是	String	预调配模板ID

请求参数

表 1-797 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-798 响应 Body 参数

参数	参数类型	描述
template_id	String	**参数说明**：预调配模板ID。
template_name	String	**参数说明**：预调配模板名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、下划线（_）、连接符（-）的组合。
description	String	**参数说明**：预调配模板的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
template_body	**ProvisioningTemplateBody** object	**参数说明**：预调配模板详细内容，json格式。
create_time	String	在物联网平台创建预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

参数	参数类型	描述
update_time	String	在物联网平台更新预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-799 ProvisioningTemplateBody

参数	参数类型	描述
parameters	Object	**参数说明**：预调配模板参数，，配置格式为{"parameter":{"type":"String"}}，其中parameter目前支持从预调配设备的证书的使用者字段提取内容，证书必须包含模板中定义的所有参数值，华为云IoT平台定义了可在预调配模板中声明和引用的如下参数: • iotda::certificate::country (国家/地区,C) • iotda::certificate::organization (组织,O) • iotda::certificate::organizational_unit (组织单位,OU) • iotda::certificate::distinguished_name_qualifier (可辨别名称限定符,dnQualifier) • iotda::certificate::state_name (省市,ST) • iotda::certificate::common_name (公用名,CN) • iotda::certificate::serial_number (序列号,serialNumber) type描述parameter的类型，目前仅支持string。 配置样例： '{"iotda::certificate::country":{"type":"String"}, "iotda::certificate::organization":{"type":"String"}, "iotda::certificate::organizational_unit":{"type":"String"}, "iotda::certificate::distinguished_name_qualifier":{"type":"String"}, "iotda::certificate::state_name":{"type":"String"}, "iotda::certificate::common_name":{"type":"String"}, "iotda::certificate::serial_number":{"type":"String"}}'
resources	**TemplateResource** object	预调配模板设备资源结构体。

表 1-800 TemplateResource

参数	参数类型	描述
device	**DeviceResource** object	预调配模板设备资源详情结构体。
policy	**PolicyResource** object	预调配模板设备策略资源详情结构体。

表 1-801 DeviceResource

参数	参数类型	描述
device_name	**ParameterRef** object	设备名称
node_id	**ParameterRef** object	设备标识码
product_id	Object	**参数说明**：设备所属的产品id，可以是一个明确的静态字符串id，也可以是动态的模板参数引用 • 明确的静态字符串："642bf260f2f9030e44210d8d"。**取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。" • 参数引用: {"ref" : "iotda::certificate::country"}
tags	Array of **TagRef** objects	**参数说明**：设备绑定的标签列表

表 1-802 ParameterRef

参数	参数类型	描述
ref	String	参数引用名称

表 1-803 TagRef

参数	参数类型	描述
tag_key	Object	**参数说明**：标签键名称，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagKey"。**取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合 ● 参数引用: {"ref" : "iotda::certificate::country"}
tag_value	Object	**参数说明**：标签值，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagValue"。**取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。 ● 参数引用: {"ref" : "iotda::certificate::country"}

表 1-804 PolicyResource

参数	参数类型	描述
policy_ids	Array of strings	**参数说明**：设备需要绑定的策略id列表

请求示例

查询指定id的预调配模板详情

```
GET https://{endpoint}/v5/iot/{project_id}/provisioning-templates/{template_id}
```

响应示例

状态码： 200

OK

```
{
  "template_id" : "5c90fa7d3c4e4405e8525079",
  "template_name" : "myTemplate",
  "description" : "myTemplate",
  "template_body" : {
    "parameters" : {
      "iotda::certificate::country" : {
        "type" : "String"
      },
      "iotda::certificate::organization" : {
        "type" : "String"
      },
      "iotda::certificate::organizational_unit" : {
        "type" : "String"
      },
```



```
    "iotda::certificate::distinguished_name_qualifier" : {
      "type" : "String"
    },
    "iotda::certificate::state_name" : {
      "type" : "String"
    },
    "iotda::certificate::common_name" : {
      "type" : "String"
    },
    "iotda::certificate::serial_number" : {
      "type" : "String"
    }
  },
  "resources" : {
    "device" : {
      "device_name" : {
        "ref" : "iotda::certificate::organization"
      },
      "node_id" : {
        "ref" : "iotda::certificate::common_name"
      },
      "product_id" : {
        "ref" : "iotda::certificate::organization"
      }
    },
    "policy" : {
      "policy_ids" : [ "5c90fa7d3c4e4405e8525079" ]
    }
  }
 },
 "create_time" : "20230810T070547Z",
 "update_time" : "20230810T070547Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowProvisioningTemplateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
```

```
            .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
ak/sk authentication scenarios
            .withAk(ak)
            .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
            .withCredential(auth)
            // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
            .withRegion(new Region("cn-north-4", iotdaEndpoint))
            .build();
        ShowProvisioningTemplateRequest request = new ShowProvisioningTemplateRequest();
        try {
            ShowProvisioningTemplateResponse response = client.showProvisioningTemplate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowProvisioningTemplateRequest()
        response = client.show_provisioning_template(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
```

```
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowProvisioningTemplateRequest{}
    response, err := client.ShowProvisioningTemplate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
404	Not Found

错误码

请参见**错误码**。

1.4.25.5 更新指定 id 的预调配模板信息

功能介绍

应用服务器可调用此接口在物联网平台更新指定id的预调配模板。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/provisioning-templates/{template_id}

表 1-805 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
template_id	是	String	预调配模板ID

请求参数

表 1-806 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-807 请求 Body 参数

参数	是否必选	参数类型	描述
description	否	String	**参数说明**：预调配模板的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
template_body	否	**ProvisioningTemplateBody** object	**参数说明**：预调配模板详细内容，json格式。

表 1-808 ProvisioningTemplateBody

参数	是否必选	参数类型	描述
parameters	是	Object	**参数说明**：预调配模板参数，，配置格式为{"parameter":{"type":"String"}}，其中parameter目前支持从预调配设备的证书的使用者字段提取内容，证书必须包含模板中定义的所有参数值，华为云IoT平台定义了可在预调配模板中声明和引用的如下参数: ● iotda::certificate::country (国家/地区,C) ● iotda::certificate::organization (组织,O) ● iotda::certificate::organizational_unit (组织单位,OU) ● iotda::certificate::distinguished_name_qualifier (可辨别名称限定符,dnQualifier) ● iotda::certificate::state_name (省市,ST) ● iotda::certificate::common_name (公用名,CN) ● iotda::certificate::serial_number (序列号,serialNumber) type描述parameter的类型，目前仅支持string。 配置样例： '{"iotda::certificate::country":{"type":"String"}, "iotda::certificate::organization":{"type":"String"}, "iotda::certificate::organizational_unit":{"type":"String"}, "iotda::certificate::distinguished_name_qualifier":{"type":"String"}, "iotda::certificate::state_name":{"type":"String"}, "iotda::certificate::common_name":{"type":"String"}, "iotda::certificate::serial_number":{"type":"String"}}'

参数	是否必选	参数类型	描述
resources	是	**TemplateResource** object	预调配模板设备资源结构体。

表 1-809 TemplateResource

参数	是否必选	参数类型	描述
device	是	**DeviceResource** object	预调配模板设备资源详情结构体。
policy	否	**PolicyResource** object	预调配模板设备策略资源详情结构体。

表 1-810 DeviceResource

参数	是否必选	参数类型	描述
device_name	否	**ParameterRef** object	设备名称
node_id	是	**ParameterRef** object	设备标识码
product_id	是	Object	**参数说明**：设备所属的产品id，可以是一个明确的静态字符串id，也可以是动态的模板参数引用 • 明确的静态字符串："642bf260f2f9030e44210d8d"。**取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。" • 参数引用: {"ref" : "iotda::certificate::country"}
tags	否	Array of **TagRef** objects	**参数说明**：设备绑定的标签列表

表 1-811 ParameterRef

参数	是否必选	参数类型	描述
ref	是	String	参数引用名称

表 1-812 TagRef

参数	是否必选	参数类型	描述
tag_key	否	Object	**参数说明**：标签键名称，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagKey"。**取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合 ● 参数引用: {"ref" : "iotda::certificate::country"}
tag_value	否	Object	**参数说明**：标签值，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagValue"。**取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。 ● 参数引用: {"ref" : "iotda::certificate::country"}

表 1-813 PolicyResource

参数	是否必选	参数类型	描述
policy_ids	否	Array of strings	**参数说明**：设备需要绑定的策略id列表

响应参数

状态码： 200

表 1-814 响应 Body 参数

参数	参数类型	描述
template_id	String	**参数说明**：预调配模板ID。
template_name	String	**参数说明**：预调配模板名称。 **取值范围**：长度不超过128，只允许中文、字母、数字、下划线（_）、连接符（-）的组合。

参数	参数类型	描述
description	String	**参数说明**：预调配模板的描述信息。 **取值范围**：长度不超过2048，只允许中文、字母、数字、以及_?'#().,&%@!-等字符的组合 最大长度：**2048**
template_body	**ProvisioningTemplateBody** object	**参数说明**：预调配模板详细内容，json格式。
create_time	String	在物联网平台创建预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。
update_time	String	在物联网平台更新预调配模板的时间。格式：yyyyMMdd'T'HHmmss'Z'，如20151212T121212Z。

表 1-815 ProvisioningTemplateBody

参数	参数类型	描述
parameters	Object	**参数说明**：预调配模板参数，，配置格式为{"parameter":{"type":"String"}}，其中parameter目前支持从预调配设备的证书的使用者字段提取内容，证书必须包含模板中定义的所有参数值，华为云IoT平台定义了可在预调配模板中声明和引用的如下参数: • iotda::certificate::country (国家/地区,C) • iotda::certificate::organization (组织,O) • iotda::certificate::organizational_unit (组织单位,OU) • iotda::certificate::distinguished_name_qualifier (可辨别名称限定符,dnQualifier) • iotda::certificate::state_name (省市,ST) • iotda::certificate::common_name (公用名,CN) • iotda::certificate::serial_number (序列号,serialNumber) type描述parameter的类型，目前仅支持string。 配置样例: '{"iotda::certificate::country":{"type":"String"}, "iotda::certificate::organization":{"type":"String"}, "iotda::certificate::organizational_unit":{"type":"String"}, "iotda::certificate::distinguished_name_qualifier":{"type":"String"}, "iotda::certificate::state_name":{"type":"String"}, "iotda::certificate::common_name":{"type":"String"}, "iotda::certificate::serial_number":{"type":"String"}}'
resources	**TemplateResource** object	预调配模板设备资源结构体。

表 1-816 TemplateResource

参数	参数类型	描述
device	**DeviceResource** object	预调配模板设备资源详情结构体。

参数	参数类型	描述
policy	**PolicyResource** object	预调配模板设备策略资源详情结构体。

表 1-817 DeviceResource

参数	参数类型	描述
device_name	**ParameterRef** object	设备名称
node_id	**ParameterRef** object	设备标识码
product_id	Object	**参数说明**：设备所属的产品id，可以是一个明确的静态字符串id，也可以是动态的模板参数引用 • 明确的静态字符串："642bf260f2f9030e44210d8d"。**取值范围**：长度不超过36，只允许字母、数字、下划线（_）、连接符（-）的组合。" • 参数引用: {"ref" : "iotda::certificate::country"}
tags	Array of **TagRef** objects	**参数说明**：设备绑定的标签列表

表 1-818 ParameterRef

参数	参数类型	描述
ref	String	参数引用名称

表 1-819 TagRef

参数	参数类型	描述
tag_key	Object	**参数说明**：标签键名称，可以是一个明确的静态字符串，也可以是动态的模板参数引用 • 明确的静态字符串："myTagKey"。**取值范围**：长度不超过64，只允许中文、字母、数字、以及_.-等字符的组合 • 参数引用: {"ref" : "iotda::certificate::country"}

参数	参数类型	描述
tag_value	Object	**参数说明**：标签值，可以是一个明确的静态字符串，也可以是动态的模板参数引用 ● 明确的静态字符串："myTagValue"。**取值范围**：长度不超过128，只允许中文、字母、数字、以及_.-等字符的组合。 ● 参数引用: {"ref" : "iotda::certificate::country"}

表 1-820 PolicyResource

参数	参数类型	描述
policy_ids	Array of strings	**参数说明**：设备需要绑定的策略id列表

请求示例

- 更新预调配模板

```
PUT https://{endpoint}/v5/iot/{project_id}/provisioning-templates/{template_id}

{
  "description" : "myTemplate",
  "template_body" : {
    "parameters" : {
      "iotda::certificate::country" : {
        "type" : "String"
      },
      "iotda::certificate::organization" : {
        "type" : "String"
      },
      "iotda::certificate::organizational_unit" : {
        "type" : "String"
      },
      "iotda::certificate::distinguished_name_qualifier" : {
        "type" : "String"
      },
      "iotda::certificate::state_name" : {
        "type" : "String"
      },
      "iotda::certificate::common_name" : {
        "type" : "String"
      },
      "iotda::certificate::serial_number" : {
        "type" : "String"
      }
    },
    "resources" : {
      "device" : {
        "device_name" : {
          "ref" : "iotda::certificate::organization"
        },
        "node_id" : {
          "ref" : "iotda::certificate::common_name"
        },
        "product_id" : {
          "ref" : "iotda::certificate::organization"
```



```
      },
      "tags" : [ {
        "tag_key" : {
          "ref" : "iotda::certificate::organization"
        },
        "tag_value" : {
          "ref" : "iotda::certificate::organizational_unit"
        }
      } ]
    },
    "policy" : {
      "policy_ids" : [ "5c90fa7d3c4e4405e8525079" ]
    }
  }
 }
}
```

- 更新预调配模板-部分参数自定义

```
PUT https://{endpoint}/v5/iot/{project_id}/provisioning-templates/{template_id}

{
 "description" : "myTemplate2",
 "template_body" : {
  "parameters" : {
    "iotda::certificate::country" : {
      "type" : "String"
    },
    "iotda::certificate::organization" : {
      "type" : "String"
    },
    "iotda::certificate::organizational_unit" : {
      "type" : "String"
    },
    "iotda::certificate::distinguished_name_qualifier" : {
      "type" : "String"
    },
    "iotda::certificate::state_name" : {
      "type" : "String"
    },
    "iotda::certificate::common_name" : {
      "type" : "String"
    },
    "iotda::certificate::serial_number" : {
      "type" : "String"
    }
  },
  "resources" : {
    "device" : {
      "device_name" : {
        "ref" : "iotda::certificate::organization"
      },
      "node_id" : {
        "ref" : "iotda::certificate::common_name"
      },
      "product_id" : "642bf260f2f9030e44210d8d",
      "tags" : [ {
        "tag_key" : "myTagKey",
        "tag_value" : "myTagValue"
      } ]
    },
    "policy" : {
      "policy_ids" : [ "5c90fa7d3c4e4405e8525079" ]
    }
  }
 }
}
```

响应示例

状态码： 200

OK

```
{
  "template_id" : "5c90fa7d3c4e4405e8525079",
  "template_name" : "myTemplate",
  "description" : "myTemplate",
  "template_body" : {
    "parameters" : {
      "iotda::certificate::country" : {
        "type" : "String"
      },
      "iotda::certificate::organization" : {
        "type" : "String"
      },
      "iotda::certificate::organizational_unit" : {
        "type" : "String"
      },
      "iotda::certificate::distinguished_name_qualifier" : {
        "type" : "String"
      },
      "iotda::certificate::state_name" : {
        "type" : "String"
      },
      "iotda::certificate::common_name" : {
        "type" : "String"
      },
      "iotda::certificate::serial_number" : {
        "type" : "String"
      }
    },
    "resources" : {
      "device" : {
        "device_name" : {
          "ref" : "iotda::certificate::organization"
        },
        "node_id" : {
          "ref" : "iotda::certificate::common_name"
        },
        "product_id" : {
          "ref" : "iotda::certificate::organization"
        }
      },
      "policy" : {
        "policy_ids" : [ "5c90fa7d3c4e4405e8525079" ]
      }
    }
  },
  "create_time" : "20230810T070547Z",
  "update_time" : "20230810T070547Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 更新预调配模板

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
```

```
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateProvisioningTemplateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateProvisioningTemplateRequest request = new UpdateProvisioningTemplateRequest();
        UpdateProvisioningTemplate body = new UpdateProvisioningTemplate();
        List<String> listPolicyPolicyIds = new ArrayList<>();
        listPolicyPolicyIds.add("5c90fa7d3c4e4405e8525079");
        PolicyResource policyResources = new PolicyResource();
        policyResources.withPolicyIds(listPolicyPolicyIds);
        List<TagRef> listDeviceTags = new ArrayList<>();
        listDeviceTags.add(
            new TagRef()
                .withTagKey("{\"ref\":\"iotda::certificate::organization\"}")
                .withTagValue("{\"ref\":\"iotda::certificate::organizational_unit\"}")
        );
        ParameterRef nodeIdDevice = new ParameterRef();
        nodeIdDevice.withRef("iotda::certificate::common_name");
        ParameterRef deviceNameDevice = new ParameterRef();
        deviceNameDevice.withRef("iotda::certificate::organization");
        DeviceResource deviceResources = new DeviceResource();
        deviceResources.withDeviceName(deviceNameDevice)
            .withNodeId(nodeIdDevice)
            .withProductId("{\"ref\":\"iotda::certificate::organization\"}")
            .withTags(listDeviceTags);
        TemplateResource resourcesTemplateBody = new TemplateResource();
        resourcesTemplateBody.withDevice(deviceResources)
            .withPolicy(policyResources);
        ProvisioningTemplateBody templateBodybody = new ProvisioningTemplateBody();
        templateBodybody.withParameters("{\"iotda::certificate::serial_number\":{\"type\":\"String\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}")
                .withResources(resourcesTemplateBody);
        body.withTemplateBody(templateBodybody);
```



```
        body.withDescription("myTemplate");
        request.withBody(body);
        try {
            UpdateProvisioningTemplateResponse response = client.updateProvisioningTemplate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 更新预调配模板-部分参数自定义

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateProvisioningTemplateSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateProvisioningTemplateRequest request = new UpdateProvisioningTemplateRequest();
        UpdateProvisioningTemplate body = new UpdateProvisioningTemplate();
        List<String> listPolicyPolicyIds = new ArrayList<>();
        listPolicyPolicyIds.add("5c90fa7d3c4e4405e8525079");
        PolicyResource policyResources = new PolicyResource();
        policyResources.withPolicyIds(listPolicyPolicyIds);
        List<TagRef> listDeviceTags = new ArrayList<>();
        listDeviceTags.add(
```

```
            new TagRef()
                .withTagKey("myTagKey")
                .withTagValue("myTagValue")
        );
        ParameterRef nodeIdDevice = new ParameterRef();
        nodeIdDevice.withRef("iotda::certificate::common_name");
        ParameterRef deviceNameDevice = new ParameterRef();
        deviceNameDevice.withRef("iotda::certificate::organization");
        DeviceResource deviceResources = new DeviceResource();
        deviceResources.withDeviceName(deviceNameDevice)
            .withNodeId(nodeIdDevice)
            .withProductId("642bf260f2f9030e44210d8d")
            .withTags(listDeviceTags);
        TemplateResource resourcesTemplateBody = new TemplateResource();
        resourcesTemplateBody.withDevice(deviceResources)
            .withPolicy(policyResources);
        ProvisioningTemplateBody templateBodybody = new ProvisioningTemplateBody();
        templateBodybody.withParameters("{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}")
            .withResources(resourcesTemplateBody);
        body.withTemplateBody(templateBodybody);
        body.withDescription("myTemplate2");
        request.withBody(body);
        try {
            UpdateProvisioningTemplateResponse response = client.updateProvisioningTemplate(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 更新预调配模板

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```



```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateProvisioningTemplateRequest()
        listPolicyIdsPolicy = [
            "5c90fa7d3c4e4405e8525079"
        ]
        policyResources = PolicyResource(
            policy_ids=listPolicyIdsPolicy
        )
        listTagsDevice = [
            TagRef(
                tag_key="{\"ref\":\"iotda::certificate::organization\"}",
                tag_value="{\"ref\":\"iotda::certificate::organizational_unit\"}"
            )
        ]
        nodeIdDevice = ParameterRef(
            ref="iotda::certificate::common_name"
        )
        deviceNameDevice = ParameterRef(
            ref="iotda::certificate::organization"
        )
        deviceResources = DeviceResource(
            device_name=deviceNameDevice,
            node_id=nodeIdDevice,
            product_id="{\"ref\":\"iotda::certificate::organization\"}",
            tags=listTagsDevice
        )
        resourcesTemplateBody = TemplateResource(
            device=deviceResources,
            policy=policyResources
        )
        templateBodybody = ProvisioningTemplateBody(
            parameters="{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}",
            resources=resourcesTemplateBody
        )
        request.body = UpdateProvisioningTemplate(
            template_body=templateBodybody,
            description="myTemplate"
        )
        response = client.update_provisioning_template(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 更新预调配模板-部分参数自定义

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
```

```
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateProvisioningTemplateRequest()
        listPolicyIdsPolicy = [
            "5c90fa7d3c4e4405e8525079"
        ]
        policyResources = PolicyResource(
            policy_ids=listPolicyIdsPolicy
        )
        listTagsDevice = [
            TagRef(
                tag_key="myTagKey",
                tag_value="myTagValue"
            )
        ]
        nodeIdDevice = ParameterRef(
            ref="iotda::certificate::common_name"
        )
        deviceNameDevice = ParameterRef(
            ref="iotda::certificate::organization"
        )
        deviceResources = DeviceResource(
            device_name=deviceNameDevice,
            node_id=nodeIdDevice,
            product_id="642bf260f2f9030e44210d8d",
            tags=listTagsDevice
        )
        resourcesTemplateBody = TemplateResource(
            device=deviceResources,
            policy=policyResources
        )
        templateBodybody = ProvisioningTemplateBody(
            parameters="{\"iotda::certificate::serial_number\":{\"type\":\"String\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}",
            resources=resourcesTemplateBody
        )
        request.body = UpdateProvisioningTemplate(
            template_body=templateBodybody,
            description="myTemplate2"
        )
        response = client.update_provisioning_template(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 更新预调配模板

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateProvisioningTemplateRequest{}
    var listPolicyIdsPolicy = []string{
        "5c90fa7d3c4e4405e8525079",
    }
    policyResources := &model.PolicyResource{
        PolicyIds: &listPolicyIdsPolicy,
    }
    tagKeyTags:= "{\"ref\":\"iotda::certificate::organization\"}"
    var tagKeyTagsInterface interface{} = tagKeyTags
    tagValueTags:= "{\"ref\":\"iotda::certificate::organizational_unit\"}"
    var tagValueTagsInterface interface{} = tagValueTags
    var listTagsDevice = []model.TagRef{
        {
            TagKey: &tagKeyTagsInterface,
            TagValue: &tagValueTagsInterface,
        },
    }
    nodeIdDevice := &model.ParameterRef{
        Ref: "iotda::certificate::common_name",
    }
    deviceNameDevice := &model.ParameterRef{
        Ref: "iotda::certificate::organization",
    }
    var productIdDevice interface{} = "{\"ref\":\"iotda::certificate::organization\"}"
    deviceResources := &model.DeviceResource{
        DeviceName: deviceNameDevice,
        NodeId: nodeIdDevice,
        ProductId: &productIdDevice,
```

```
        Tags: &listTagsDevice,
    }
    resourcesTemplateBody := &model.TemplateResource{
        Device: deviceResources,
        Policy: policyResources,
    }
    var parametersTemplateBody interface{} = "{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}"
    templateBodybody := &model.ProvisioningTemplateBody{
        Parameters: &parametersTemplateBody,
        Resources: resourcesTemplateBody,
    }
    descriptionUpdateProvisioningTemplate:= "myTemplate"
    request.Body = &model.UpdateProvisioningTemplate{
        TemplateBody: templateBodybody,
        Description: &descriptionUpdateProvisioningTemplate,
    }
    response, err := client.UpdateProvisioningTemplate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 更新预调配模板-部分参数自定义

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateProvisioningTemplateRequest{}
    var listPolicyIdsPolicy = []string{
        "5c90fa7d3c4e4405e8525079",
```



```
    }
    policyResources := &model.PolicyResource{
        PolicyIds: &listPolicyIdsPolicy,
    }
    tagKeyTags:= "myTagKey"
    var tagKeyTagsInterface interface{} = tagKeyTags
    tagValueTags:= "myTagValue"
    var tagValueTagsInterface interface{} = tagValueTags
    var listTagsDevice = []model.TagRef{
        {
            TagKey: &tagKeyTagsInterface,
            TagValue: &tagValueTagsInterface,
        },
    }
    nodeIdDevice := &model.ParameterRef{
        Ref: "iotda::certificate::common_name",
    }
    deviceNameDevice := &model.ParameterRef{
        Ref: "iotda::certificate::organization",
    }
    var productIdDevice interface{} = "642bf260f2f9030e44210d8d"
    deviceResources := &model.DeviceResource{
        DeviceName: deviceNameDevice,
        NodeId: nodeIdDevice,
        ProductId: &productIdDevice,
        Tags: &listTagsDevice,
    }
    resourcesTemplateBody := &model.TemplateResource{
        Device: deviceResources,
        Policy: policyResources,
    }
    var parametersTemplateBody interface{} = "{\"iotda::certificate::serial_number\":{\"type\":\"String
\"},\"iotda::certificate::organizational_unit\":{\"type\":\"String\"},\"iotda::certificate::state_name\":
{\"type\":\"String\"},\"iotda::certificate::distinguished_name_qualifier\":{\"type\":\"String
\"},\"iotda::certificate::country\":{\"type\":\"String\"},\"iotda::certificate::organization\":{\"type\":\"String
\"},\"iotda::certificate::common_name\":{\"type\":\"String\"}}"
    templateBodybody := &model.ProvisioningTemplateBody{
        Parameters: &parametersTemplateBody,
        Resources: resourcesTemplateBody,
    }
    descriptionUpdateProvisioningTemplate:= "myTemplate2"
    request.Body = &model.UpdateProvisioningTemplate{
        TemplateBody: templateBodybody,
        Description: &descriptionUpdateProvisioningTemplate,
    }
    response, err := client.UpdateProvisioningTemplate(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request

状态码	描述
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.26 自定义鉴权

1.4.26.1 创建自定义鉴权

功能介绍

应用服务器可调用此接口在物联网平台创建一个自定义鉴权。自定义鉴权是指用户可以通过函数服务自定义实现鉴权逻辑，以对接入平台的设备进行身份认证。

- 单个实例最大可配置10个自定义鉴权
- 仅标准版实例、企业版实例支持该接口调用，基础版不支持。

调用方法

请参见**如何调用API**。

URI

POST /v5/iot/{project_id}/device-authorizers

表 1-821 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

请求参数

表 1-822 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-823 请求 Body 参数

参数	是否必选	参数类型	描述
authorizer_name	是	String	**参数说明**：自定义鉴权器名称，同一租户下的自定义鉴权器名称不能重复。 **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。
func_urn	是	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数，即自定义鉴权器对应的处理函数地址。
signing_enable	否	Boolean	**参数说明**：是否启动签名校验，启动签名校验后不满足签名要求的鉴权信息将被拒绝，以减少无效的函数调用。推荐用户进行安全的签名校验，默认开启， 开启时signing_token与signing_public_key必填。 缺省值：**true**

参数	是否必选	参数类型	描述
signing_token	否	String	**参数说明**：签名校验的Key值，开启签名校验时使用。 **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。
signing_public_key	否	String	**参数说明**：签名校验的公钥，开启签名校验时使用。用于认证设备携带的签名信息是否正确。
default_authorizer	否	Boolean	**参数说明**：当前自定义鉴权是否为默认的鉴权方式，默认为false，当设置为true时，用户所有支持SNI的设备，如果在鉴权时不指定使用特定的设备鉴权，将统一使用当前鉴权器策略进行鉴权。 缺省值：**false**
status	否	String	**参数说明**：是否激活该鉴权方式 ● ACTIVE：该鉴权为激活状态。 ● INACTIVE：该鉴权为停用状态。 缺省值：**INACTIVE**
cache_enable	否	Boolean	**参数说明**：是否开启缓存，默认为false，设备为true时，当设备入参（username，clientId，password，以及证书信息，函数urn）不变时，当缓存结果存在时，将直接使用缓存结果，建议在调试时设置为false，生产时设置为true，避免频繁调用函数。 缺省值：**false**

响应参数

状态码： 201

表 1-824 响应 Body 参数

参数	参数类型	描述
authorizer_id	String	**参数说明**：自定义鉴权ID。

参数	参数类型	描述
authorizer_name	String	**参数说明**：自定义鉴权器名称，同一租户下的自定义鉴权器名称不能重复。 **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。
func_name	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**
func_urn	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数，即自定义鉴权器对应的处理函数地址。 最小长度：**0** 最大长度：**65535**
signing_enable	Boolean	**参数说明**：是否启动签名校验，启动签名校验后不满足签名要求的鉴权信息将被拒绝，以减少无效的函数调用。推荐用户进行安全的签名校验，默认开启。 缺省值：**true**
signing_token	String	**参数说明**：签名校验的Key值，开启签名校验时使用。 **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。
signing_public_key	String	**参数说明**：签名校验的公钥，开启签名校验时使用。用于认证设备携带的签名信息是否正确。 最小长度：**0** 最大长度：**65535**
default_authorizer	Boolean	**参数说明**：是否为默认的鉴权方式，默认为false。 缺省值：**false**
status	String	**参数说明**：是否激活该鉴权方式 ● ACTIVE：该鉴权为激活状态。 ● INACTIVE：该鉴权为停用状态。 缺省值：**INACTIVE**
cache_enable	Boolean	**参数说明**：是否开启缓存，默认为false，设备为true时，当设备入参（username，clientId，password，以及证书信息，函数urn）不变时，当缓存结果存在时，将直接使用缓存结果，建议在调试时设置为false，生产时设置为true，避免频繁调用函数。 缺省值：**false**

参数	参数类型	描述
create_time	String	在物联网平台进行自定义鉴权相关操作的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。
update_time	String	在物联网平台更新自定义鉴权相关操作的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。

请求示例

创建自定义鉴权

```
POST https://{endpoint}/v5/iot/{project_id}/device-authorizers

{
  "authorizer_name" : "myTest",
  "func_urn" : "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
  "signing_enable" : true,
  "signing_token" : "string",
  "signing_public_key" : "string",
  "default_authorizer" : false,
  "status" : "ACTIVE",
  "cache_enable" : true
}
```

响应示例

状态码： 201

Created

```
{
  "authorizer_id" : "5c90fa7d3c4e4405e8525079",
  "authorizer_name" : "myTest",
  "func_name" : "mqtt_auth",
  "func_urn" : "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
  "signing_enable" : true,
  "signing_token" : "string",
  "signing_public_key" : "string",
  "default_authorizer" : false,
  "status" : "ACTIVE",
  "cache_enable" : false,
  "create_time" : "20231031T070547Z",
  "update_time" : "20231031T070547Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建自定义鉴权

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
```



```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class CreateDeviceAuthorizerSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        CreateDeviceAuthorizerRequest request = new CreateDeviceAuthorizerRequest();
        CreateDeviceAuthorizer body = new CreateDeviceAuthorizer();
        body.withCacheEnable(true);
        body.withStatus("ACTIVE");
        body.withDefaultAuthorizer(false);
        body.withSigningPublicKey("string");
        body.withSigningToken("string");
        body.withSigningEnable(true);
        body.withFuncUrn("urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest");
        body.withAuthorizerName("myTest");
        request.withBody(body);
        try {
            CreateDeviceAuthorizerResponse response = client.createDeviceAuthorizer(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建自定义鉴权

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = CreateDeviceAuthorizerRequest()
        request.body = CreateDeviceAuthorizer(
            cache_enable=True,
            status="ACTIVE",
            default_authorizer=False,
            signing_public_key="string",
            signing_token="string",
            signing_enable=True,
            func_urn="urn:fss:cn-
north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
            authorizer_name="myTest"
        )
        response = client.create_device_authorizer(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建自定义鉴权

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
```

```
variables and decrypted during use to ensure security.
   // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
   ak := os.Getenv("CLOUD_SDK_AK")
   sk := os.Getenv("CLOUD_SDK_SK")
   // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
   endpoint := "<YOUR ENDPOINT>"

   auth := basic.NewCredentialsBuilder().
      WithAk(ak).
      WithSk(sk).
      // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
      WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
      Build()

   client := iotda.NewIoTDAClient(
      iotda.IoTDAClientBuilder().
         // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
         WithRegion(region.NewRegion("cn-north-4", endpoint)).
         WithCredential(auth).
         Build())

   request := &model.CreateDeviceAuthorizerRequest{}
   cacheEnableCreateDeviceAuthorizer:= true
   statusCreateDeviceAuthorizer:= "ACTIVE"
   defaultAuthorizerCreateDeviceAuthorizer:= false
   signingPublicKeyCreateDeviceAuthorizer:= "string"
   signingTokenCreateDeviceAuthorizer:= "string"
   signingEnableCreateDeviceAuthorizer:= true
   request.Body = &model.CreateDeviceAuthorizer{
      CacheEnable: &cacheEnableCreateDeviceAuthorizer,
      Status: &statusCreateDeviceAuthorizer,
      DefaultAuthorizer: &defaultAuthorizerCreateDeviceAuthorizer,
      SigningPublicKey: &signingPublicKeyCreateDeviceAuthorizer,
      SigningToken: &signingTokenCreateDeviceAuthorizer,
      SigningEnable: &signingEnableCreateDeviceAuthorizer,
      FuncUrn: "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
      AuthorizerName: "myTest",
   }
   response, err := client.CreateDeviceAuthorizer(request)
   if err == nil {
      fmt.Printf("%+v\n", response)
   } else {
      fmt.Println(err)
   }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden

状态码	描述
500	Internal Server Error

错误码

请参见**错误码**。

1.4.26.2 查询自定义鉴权列表

功能介绍

应用服务器可调用此接口在物联网平台查询自定义鉴权列表。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-authorizers

表 1-825 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。

表 1-826 Query 参数

参数	是否必选	参数类型	描述
authorizer_name	否	String	**参数说明**：自定义鉴权名称，同一租户下的自定义鉴权器名称不能重复。 **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。
limit	否	Integer	**参数说明**：分页查询时每页显示的记录数。 **取值范围**：1-50的整数，默认值为10。 最小值：**1** 最大值：**50** 缺省值：**10**

参数	是否必选	参数类型	描述
marker	否	String	**参数说明**：上一次分页查询结果中最后一条记录的ID，在上一次分页查询时由物联网平台返回获得。分页查询时物联网平台是按marker也就是记录ID降序查询的，越新的数据记录ID也会越大。若填写marker，则本次只查询记录ID小于marker的数据记录。若不填写，则从记录ID最大也就是最新的一条数据开始查询。如果需要依次查询所有数据，则每次查询时必须填写上一次查询响应中的marker值。 **取值范围**：长度为24的十六进制字符串，默认值为ffffffffffffffffffffffff。 缺省值：**ffffffffffffffffffffffff**
offset	否	Integer	**参数说明**：表示从marker后偏移offset条记录开始查询。默认为0，取值范围为0-500的整数。当offset为0时，表示从marker后第一条记录开始输出。限制offset最大值是出于API性能考虑，您可以搭配marker使用该参数实现翻页，例如每页50条记录，1-11页内都可以直接使用offset跳转到指定页，但到11页后，由于offset限制为500，您需要使用第11页返回的marker作为下次查询的marker，以实现翻页到12-22页。 **取值范围**：0-500的整数，默认为0。 最小值：**0** 最大值：**500** 缺省值：**0**

请求参数

表 1-827 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-828 响应 Body 参数

参数	参数类型	描述
authorizers	Array of **DeviceAuthorizerSimple** objects	**参数说明**：自定义鉴权列表。
page	**Page** object	查询结果的分页信息。

表 1-829 DeviceAuthorizerSimple

参数	参数类型	描述
authorizer_id	String	**参数说明**：自定义鉴权ID。
authorizer_name	String	**参数说明**：自定义鉴权器名称，同一租户下的自定义鉴权器名称不能重复。 **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。
func_name	String	**参数说明**：自定义鉴权器对应的函数名称。

参数	参数类型	描述
func_urn	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数，即自定义鉴权器对应的处理函数地址。
signing_enable	Boolean	**参数说明**：是否启动签名校验，启动签名校验后不满足签名要求的鉴权信息将被拒绝，以减少无效的函数调用。推荐用户进行安全的签名校验，默认开启。 缺省值：**true**
default_authorizer	Boolean	**参数说明**：当前自定义鉴权是否为默认的鉴权方式，默认为false，当设置为true时，用户所有支持SNI的设备，如果在鉴权时不指定使用特定的设备鉴权，将统一使用当前鉴权器策略进行鉴权。 缺省值：**false**
status	String	**参数说明**：是否激活该鉴权方式 ● ACTIVE：该鉴权为激活状态。 ● INACTIVE：该鉴权为停用状态。 缺省值：**INACTIVE**
cache_enable	Boolean	**参数说明**：是否开启缓存，默认为false，设备为true时，当设备入参（username，clientId，password，以及证书信息，函数urn）不变时，当缓存结果存在时，将直接使用缓存结果，建议在调试时设置为false，生产时设置为true，避免频繁调用函数。 缺省值：**false**
create_time	String	在物联网平台查询自定义鉴权的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。
update_time	String	在物联网平台更新查询自定义鉴权的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。

表 1-830 Page

参数	参数类型	描述
count	Long	满足查询条件的记录总数。
marker	String	本次分页查询结果中最后一条记录的ID，可在下一次分页查询时使用。

请求示例

查询租户下的自定义鉴权列表

```
GET https://{endpoint}/v5/iot/{project_id}/device-authorizers
```

响应示例

状态码： 200

OK

```
{
  "authorizers" : [ {
    "authorizer_id" : "5c90fa7d3c4e4405e8525079",
    "authorizer_name" : "myTest",
    "func_name" : "mqtt_auth",
    "func_urn" : "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:lates",
    "signing_enable" : true,
    "default_authorizer" : false,
    "status" : "ACTIVE",
    "cache_enable" : false,
    "create_time" : "20231031T070547Z",
    "update_time" : "20231031T070547Z"
  } ],
  "page" : {
    "count" : 1,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ListDeviceAuthorizersSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative
```

```
ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如
"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ListDeviceAuthorizersRequest request = new ListDeviceAuthorizersRequest();
        request.withAuthorizerName("<authorizer_name>");
        request.withLimit(<limit>);
        request.withMarker("<marker>");
        request.withOffset(<offset>);
        try {
            ListDeviceAuthorizersResponse response = client.listDeviceAuthorizers(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ListDeviceAuthorizersRequest()
        request.authorizer_name = "<authorizer_name>"
        request.limit = <limit>
        request.marker = "<marker>"
```



```
        request.offset = <offset>
        response = client.list_device_authorizers(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ListDeviceAuthorizersRequest{}
    authorizerNameRequest:= "<authorizer_name>"
    request.AuthorizerName = &authorizerNameRequest
    limitRequest:= int32(<limit>)
    request.Limit = &limitRequest
    markerRequest:= "<marker>"
    request.Marker = &markerRequest
    offsetRequest:= int32(<offset>)
    request.Offset = &offsetRequest
    response, err := client.ListDeviceAuthorizers(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK

错误码

请参见**错误码**。

1.4.26.3 删除自定义鉴权

功能介绍

应用服务器可调用此接口在物联网平台上删除指定自定义鉴权。

调用方法

请参见**如何调用API**。

URI

DELETE /v5/iot/{project_id}/device-authorizers/{authorizer_id}

表 1-831 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
authorizer_id	是	String	自定义鉴权ID

请求参数

表 1-832 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

无

请求示例

删除指定id的自定义鉴权

```
DELETE https://{endpoint}/v5/iot/{project_id}/device-authorizers/{authorizer_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;
```



```
public class DeleteDeviceAuthorizerSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        DeleteDeviceAuthorizerRequest request = new DeleteDeviceAuthorizerRequest();
        try {
            DeleteDeviceAuthorizerResponse response = client.deleteDeviceAuthorizer(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())
```

```
    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = DeleteDeviceAuthorizerRequest()
        response = client.delete_device_authorizer(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk
authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.DeleteDeviceAuthorizerRequest{}
    response, err := client.DeleteDeviceAuthorizer(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	No Content
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.4.26.4 查询自定义鉴权详情

功能介绍

应用服务器可调用此接口在物联网平台查询指定自定义鉴权ID的详细信息。

调用方法

请参见**如何调用API**。

URI

GET /v5/iot/{project_id}/device-authorizers/{authorizer_id}

表 1-833 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
authorizer_id	是	String	自定义鉴权ID

请求参数

表 1-834 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

响应参数

状态码： 200

表 1-835 响应 Body 参数

参数	参数类型	描述
authorizer_id	String	**参数说明**：自定义鉴权ID。
authorizer_name	String	**参数说明**：自定义鉴权器名称，同一租户下的自定义鉴权器名称不能重复。 **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。
func_name	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**
func_urn	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数，即自定义鉴权器对应的处理函数地址。 最小长度：**0** 最大长度：**65535**

参数	参数类型	描述
signing_enable	Boolean	**参数说明**：是否启动签名校验，启动签名校验后不满足签名要求的鉴权信息将被拒绝，以减少无效的函数调用。推荐用户进行安全的签名校验，默认开启。 缺省值：**true**
signing_token	String	**参数说明**：签名校验的Key值，开启签名校验时使用。 **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。
signing_public_key	String	**参数说明**：签名校验的公钥，开启签名校验时使用。用于认证设备携带的签名信息是否正确。 最小长度：**0** 最大长度：**65535**
default_authorizer	Boolean	**参数说明**：是否为默认的鉴权方式，默认为false。 缺省值：**false**
status	String	**参数说明**：是否激活该鉴权方式 ● ACTIVE：该鉴权为激活状态。 ● INACTIVE：该鉴权为停用状态。 缺省值：**INACTIVE**
cache_enable	Boolean	**参数说明**：是否开启缓存，默认为false，设备为true时，当设备入参（username，clientId，password，以及证书信息，函数urn）不变时，当缓存结果存在时，将直接使用缓存结果，建议在调试时设置为false，生产时设置为true，避免频繁调用函数。 缺省值：**false**
create_time	String	在物联网平台进行自定义鉴权相关操作的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。
update_time	String	在物联网平台更新自定义鉴权相关操作的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。

请求示例

查询指定id的自定义鉴权详情

```
GET https://{endpoint}/v5/iot/{project_id}/device-authorizers/{authorizer_id}
```

响应示例

状态码： 200

OK

```
{
  "authorizer_id" : "5c90fa7d3c4e4405e8525079",
  "authorizer_name" : "myTest",
  "func_name" : "mqtt_auth",
  "func_urn" : "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
  "signing_enable" : true,
  "signing_token" : "string",
  "signing_public_key" : "string",
  "default_authorizer" : false,
  "status" : "ACTIVE",
  "cache_enable" : false,
  "create_time" : "20231031T070547Z",
  "update_time" : "20231031T070547Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class ShowDeviceAuthorizerSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        ShowDeviceAuthorizerRequest request = new ShowDeviceAuthorizerRequest();
        try {
            ShowDeviceAuthorizerResponse response = client.showDeviceAuthorizer(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
```

```
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak,
sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象
如：  .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = ShowDeviceAuthorizerRequest()
        response = client.show_device_authorizer(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
```

```
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.ShowDeviceAuthorizerRequest{}
    response, err := client.ShowDeviceAuthorizer(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
404	Not Found

错误码

请参见**错误码**。

1.4.26.5 更新指定 id 的自定义鉴权

功能介绍

应用服务器可调用此接口在物联网平台更新指定id的自定义鉴权。

调用方法

请参见**如何调用API**。

URI

PUT /v5/iot/{project_id}/device-authorizers/{authorizer_id}

表 1-836 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	**参数说明**：项目ID。获取方法请参见 **获取项目ID** 。
authorizer_id	是	String	自定义鉴权ID

请求参数

表 1-837 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	否	String	**参数说明**：用户Token。通过调用IAM服务 **获取IAM用户Token** 接口获取，接口返回的响应消息头中“X-Subject-Token”就是需要获取的用户Token。简要的获取方法样例请参见 **Token认证**。
Instance-Id	否	String	**参数说明**：实例ID。物理多租下各实例的唯一标识，建议携带该参数，在使用专业版时必须携带该参数。您可以在IoTDA管理控制台界面，选择左侧导航栏“总览”页签查看当前实例的ID，具体获取方式请参考**查看实例详情** 。

表 1-838 请求 Body 参数

参数	是否必选	参数类型	描述
authorizer_name	否	String	**参数说明**：自定义鉴权器名称，同一租户下的自定义鉴权器名称不能重复。 **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。

参数	是否必选	参数类型	描述
func_urn	否	String	**参数说明**：函数的URN（Uniform Resource Name），唯一标识函数，即自定义鉴权器对应的处理函数地址。
signing_enable	否	Boolean	**参数说明**：是否启动签名校验，启动签名校验后不满足签名要求的鉴权信息将被拒绝，以减少无效的函数调用。推荐用户进行安全的签名校验，默认开启，开启时signing_token与signing_public_key必填。
signing_token	否	String	**参数说明**：签名校验的Key值，开启签名校验时使用。 **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。
signing_public_key	否	String	**参数说明**：签名校验的公钥，开启签名校验时使用。用于认证设备携带的签名信息是否正确。
default_authorizer	否	Boolean	**参数说明**：当前自定义鉴权是否为默认的鉴权方式，默认为false，当设置为true时，用户所有支持SNI的设备，如果在鉴权时不指定使用特定的设备鉴权，将统一使用当前鉴权器策略进行鉴权。
status	否	String	**参数说明**：是否激活该鉴权方式，默认为激活。 ● ACTIVE：该鉴权为激活状态。 ● INACTIVE：该鉴权为停用状态。
cache_enable	否	Boolean	**参数说明**：是否开启缓存，默认为false，设备为true时，当设备入参（username，clientId，password，以及证书信息，函数urn）不变时，当缓存结果存在时，将直接使用缓存结果，建议在调试时设置为false，生产时设置为true，避免频繁调用函数。

响应参数

状态码： 200

表 1-839 响应 Body 参数

参数	参数类型	描述
authorizer_id	String	**参数说明**：自定义鉴权ID。
authorizer_name	String	**参数说明**：自定义鉴权器名称，同一租户下的自定义鉴权器名称不能重复。 **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。
func_name	String	**参数说明**：函数名称。 最小长度：**0** 最大长度：**65535**
func_urn	String	**参数说明**：函数的URN（ Uniform Resource Name ），唯一标识函数，即自定义鉴权器对应的处理函数地址。 最小长度：**0** 最大长度：**65535**
signing_enable	Boolean	**参数说明**：是否启动签名校验，启动签名校验后不满足签名要求的鉴权信息将被拒绝，以减少无效的函数调用。推荐用户进行安全的签名校验，默认开启。 缺省值：**true**
signing_token	String	**参数说明**：签名校验的Key值，开启签名校验时使用。 **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。
signing_public_key	String	**参数说明**：签名校验的公钥，开启签名校验时使用。用于认证设备携带的签名信息是否正确。 最小长度：**0** 最大长度：**65535**
default_authorizer	Boolean	**参数说明**：是否为默认的鉴权方式，默认为false。 缺省值：**false**
status	String	**参数说明**：是否激活该鉴权方式 ● ACTIVE：该鉴权为激活状态。 ● INACTIVE：该鉴权为停用状态。 缺省值：**INACTIVE**

参数	参数类型	描述
cache_enable	Boolean	**参数说明**：是否开启缓存，默认为false，设备为true时，当设备入参（username，clientId，password，以及证书信息，函数urn）不变时，当缓存结果存在时，将直接使用缓存结果，建议在调试时设置为false，生产时设置为true，避免频繁调用函数。 缺省值：**false**
create_time	String	在物联网平台进行自定义鉴权相关操作的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。
update_time	String	在物联网平台更新自定义鉴权相关操作的时间。格式：yyyyMMdd'T'HHmmss'Z'，如：20151212T121212Z。

请求示例

更新自定义鉴权

```
PUT https://{endpoint}/v5/iot/{project_id}/device-authorizers/{authorizer_id}

{
  "authorizer_name" : "myTest",
  "func_urn" : "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
  "signing_enable" : true,
  "signing_token" : "string",
  "signing_public_key" : "string",
  "default_authorizer" : false,
  "status" : "ACTIVE",
  "cache_enable" : false
}
```

响应示例

状态码： 200

OK

```
{
  "authorizer_id" : "5c90fa7d3c4e4405e8525079",
  "authorizer_name" : "myTest",
  "func_name" : "mqtt_auth",
  "func_urn" : "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
  "signing_enable" : true,
  "signing_token" : "string",
  "signing_public_key" : "string",
  "default_authorizer" : false,
  "status" : "ACTIVE",
  "cache_enable" : false,
  "create_time" : "20231031T070547Z",
  "update_time" : "20231031T070947Z"
}
```

SDK 代码示例

SDK代码示例如下。

Java

更新自定义鉴权

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.AbstractCredentials;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.core.region.Region;
import com.huaweicloud.sdk.iotda.v5.*;
import com.huaweicloud.sdk.iotda.v5.model.*;


public class UpdateDeviceAuthorizerSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
        String iotdaEndpoint = "<YOUR ENDPOINT>";

        ICredential auth = new BasicCredentials()
                // 标准版/企业版需要使用衍生算法，基础版请删除配置"withDerivedPredicate";
                .withDerivedPredicate(AbstractCredentials.DEFAULT_DERIVED_PREDICATE) // Used in derivative ak/sk authentication scenarios
                .withAk(ak)
                .withSk(sk);

        IoTDAClient client = IoTDAClient.newBuilder()
                .withCredential(auth)
                // 标准版/企业版：需自行创建Region对象，基础版：请使用IoTDARegion的region对象，如"withRegion(IoTDARegion.CN_NORTH_4)"
                .withRegion(new Region("cn-north-4", iotdaEndpoint))
                .build();
        UpdateDeviceAuthorizerRequest request = new UpdateDeviceAuthorizerRequest();
        UpdateDeviceAuthorizer body = new UpdateDeviceAuthorizer();
        body.withCacheEnable(false);
        body.withStatus("ACTIVE");
        body.withDefaultAuthorizer(false);
        body.withSigningPublicKey("string");
        body.withSigningToken("string");
        body.withSigningEnable(true);
        body.withFuncUrn("urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest");
        body.withAuthorizerName("myTest");
        request.withBody(body);
        try {
            UpdateDeviceAuthorizerResponse response = client.updateDeviceAuthorizer(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
```



```
    }
}
```

Python

更新自定义鉴权

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkcore.auth.credentials import DerivedCredentials
from huaweicloudsdkcore.region.region import Region as coreRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkiotda.v5 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    // ENDPOINT：请在控制台的"总览"界面的"平台接入地址"中查看“应用侧”的https接入地址。
    iotdaEndpoint = "<YOUR ENDPOINT>";

    credentials = BasicCredentials(ak, sk).with_derived_predicate(DerivedCredentials.get_default_derived_predicate())

    client = IoTDAClient.new_builder() \
        .with_credentials(credentials) \
        # 标准版/企业版：需要使用自行创建的Region对象，基础版：请选择IoTDAClient中的Region对象 如： .with_region(IoTDARegion.CN_NORTH_4)
        .with_region(coreRegion(id="cn-north-4", endpoint=endpoint)) \
        .build()

    try:
        request = UpdateDeviceAuthorizerRequest()
        request.body = UpdateDeviceAuthorizer(
            cache_enable=False,
            status="ACTIVE",
            default_authorizer=False,
            signing_public_key="string",
            signing_token="string",
            signing_enable=True,
            func_urn="urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest",
            authorizer_name="myTest"
        )
        response = client.update_device_authorizer(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

更新自定义鉴权

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    iotda "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5"
```

```
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/iotda/v5/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/region"
    core_auth "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    // endpoint：请在控制台的"总览"界面的"平台接入地址"中查看"应用侧"的https接入地址
    endpoint := "<YOUR ENDPOINT>"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        // 企业版/标准版需要使用衍生算法，基础版请删除该配置"WithDerivedPredicate"
        WithDerivedPredicate(core_auth.GetDefaultDerivedPredicate()). // Used in derivative ak/sk authentication scenarios
        Build()

    client := iotda.NewIoTDAClient(
        iotda.IoTDAClientBuilder().
            // 标准版/企业版需要自行创建region，基础版使用IoTDARegion中的region对象
            WithRegion(region.NewRegion("cn-north-4", endpoint)).
            WithCredential(auth).
            Build())

    request := &model.UpdateDeviceAuthorizerRequest{}
    cacheEnableUpdateDeviceAuthorizer:= false
    statusUpdateDeviceAuthorizer:= "ACTIVE"
    defaultAuthorizerUpdateDeviceAuthorizer:= false
    signingPublicKeyUpdateDeviceAuthorizer:= "string"
    signingTokenUpdateDeviceAuthorizer:= "string"
    signingEnableUpdateDeviceAuthorizer:= true
    funcUrnUpdateDeviceAuthorizer:= "urn:fss:cn-north-5:d92d9c5eb8e347b5bb31ecfe5bc0c4e1:function:default:mqtt_auth:latest"
    authorizerNameUpdateDeviceAuthorizer:= "myTest"
    request.Body = &model.UpdateDeviceAuthorizer{
        CacheEnable: &cacheEnableUpdateDeviceAuthorizer,
        Status: &statusUpdateDeviceAuthorizer,
        DefaultAuthorizer: &defaultAuthorizerUpdateDeviceAuthorizer,
        SigningPublicKey: &signingPublicKeyUpdateDeviceAuthorizer,
        SigningToken: &signingTokenUpdateDeviceAuthorizer,
        SigningEnable: &signingEnableUpdateDeviceAuthorizer,
        FuncUrn: &funcUrnUpdateDeviceAuthorizer,
        AuthorizerName: &authorizerNameUpdateDeviceAuthorizer,
    }
    response, err := client.UpdateDeviceAuthorizer(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	OK
400	Bad Request
403	Forbidden
404	Not Found
500	Internal Server Error

错误码

请参见**错误码**。

1.5 权限与授权项

1.5.1 权限与授权说明

如果您需要对您所拥有的设备接入服务（IoTDA）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），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，不影响您使用设备接入服务（IoTDA）的其它功能。

默认情况下，新建的IAM用户没有任何权限，您需要将其加入用户组，并给用户组授予策略或角色，才能使用户组中的用户获得相应的权限，这一过程称为授权。授权后，用户就可以基于已有权限对云服务进行操作。

权限根据授权的精细程度，分为**角色**和**策略**。角色以服务为粒度，是IAM最初提供的一种根据用户的工作职能定义权限的粗粒度授权机制。策略以API接口为粒度进行权限拆分，授权更加精细，可以精确到某个操作、资源和条件，能够满足企业对权限最小化的安全管控要求。

说明

如果您要允许或是禁止某个接口的操作权限，请使用策略。

账号具备所有接口的调用权限，如果使用账号下的IAM用户发起API请求时，该IAM用户必须具备调用该接口所需的权限，否则，API请求将调用失败。每个接口所需要的权限，与各个接口所对应的授权项相对应，只有发起请求的用户被授予授权项所对应的策略，该用户才能成功调用该接口。例如，用户要调用接口来创建设备，那么这个IAM用户被授予的策略中必须包含允许“iotda:devices:register”的授权项，该接口才能调用成功。

支持的授权项

策略包含系统策略和自定义策略，如果系统策略不满足授权要求，管理员可以创建自定义策略，并通过给用户组授予自定义策略来进行精细的访问控制。策略支持的操作与API相对应，授权项列表说明如下：

- API分组：接口所属分组。
- 权限：允许或拒绝某项操作。
- 对应API接口：自定义策略实际调用的API接口。
- 授权项：自定义策略中支持的Action，在自定义策略中的Action中写入授权项，可以实现授权项对应的权限功能。

1.5.2 授权项列表

API分组	权限	API	授权项（Action）
产品管理	创建产品	POST /v5/iot/{project_id}/products	iotda:products:create
	查询产品列表	GET /v5/iot/{project_id}/products	iotda:products:queryList
	查询产品	GET /v5/iot/{project_id}/products/{product_id}	iotda:products:query
	修改产品	PUT /v5/iot/{project_id}/products/{product_id}	iotda:products:modify
	删除产品	DELETE /v5/iot/{project_id}/products/{product_id}	iotda:products:delete
设备管理	创建设备	POST /v5/iot/{project_id}/devices	iotda:devices:register
	查询设备列表	GET /v5/iot/{project_id}/devices	iotda:devices:queryList
	查询设备	GET /v5/iot/{project_id}/devices/{device_id}	iotda:devices:query
	修改设备	PUT /v5/iot/{project_id}/devices/{device_id}	iotda:devices:modify
	删除设备	DELETE /v5/iot/{project_id}/devices/{device_id}	iotda:devices:delete
	重置设备密钥	POST /v5/iot/{project_id}/devices/{device_id}/action	iotda:devices:resetSecret
	冻结设备	POST /v5/iot/{project_id}/devices/{device_id}/freeze	iotda:devices:freeze
	解冻设备	POST /v5/iot/{project_id}/devices/{device_id}/unfreeze	iotda:devices:unfreeze
	重置设备指纹	POST /v5/iot/{project_id}/devices/{device_id}/reset-fingerprint	iotda:devices:resetFingerprint
	灵活搜索设备列表	POST /v5/iot/{project_id}/search/query-devices	iotda:devices:queryList

设备消息管理	下发设备消息	POST /v5/iot/{project_id}/devices/{device_id}/messages	iotda:messages:send
	查询设备消息	GET /v5/iot/{project_id}/devices/{device_id}/messages	iotda:messages:queryList
	查询指定消息id的消息	GET /v5/iot/{project_id}/devices/{device_id}/messages/{message_id}	iotda:messages:query
	下发广播消息	POST /v5/iot/{project_id}/broadcast-messages	iotda:message:broadcast
设备命令管理	下发设备命令	POST /v5/iot/{project_id}/devices/{device_id}/commands	iotda:commands:send
	下发异步设备命令	POST /v5/iot/{project_id}/devices/{device_id}/async-commands	iotda:asynccommands:send
	查询指定id的命令	GET /v5/iot/{project_id}/devices/{device_id}/async-commands/{command_id}	iotda:asynccommands:query
设备属性管理	修改设备属性	PUT /v5/iot/{project_id}/devices/{device_id}/properties	iotda:properties:modify
	查询设备属性	GET /v5/iot/{project_id}/devices/{device_id}/properties	iotda:properties:query
设备影子	查询设备影子数据	GET /v5/iot/{project_id}/devices/{device_id}/shadow	iotda:shadow:query
	配置设备影子预期数据	PUT /v5/iot/{project_id}/devices/{device_id}/shadow	iotda:shadow:config
AMQP队列管理	创建AMQP队列	POST /v5/iot/{project_id}/amqp-queues	iotda:amqpqueue:create
	查询AMQP列表	GET /v5/iot/{project_id}/amqp-queues	iotda:amqpqueue:queryList
	查询单个AMQP队列	GET /v5/iot/{project_id}/amqp-queues/{queue_id}	iotda:amqpqueue:query
	删除AMQP队列	DELETE /v5/iot/{project_id}/amqp-queues/{queue_id}	iotda:amqpqueue:delete
接入码管理	生成接入凭证	POST /v5/iot/{project_id}/auth/accesscode	iotda:accesscode:create
流转规则管理	创建规则触发条件	POST /v5/iot/{project_id}/routing-rule/rules	iotda:routingrules:create
	查询规则条件列表	GET /v5/iot/{project_id}/routing-rule/rules	iotda:routingrules:queryList
	查询规则条件	GET /v5/iot/{project_id}/routing-rule/rules/{rule_id}	iotda:routingrules:query

	修改规则触发条件	PUT /v5/iot/{project_id}/routing-rule/rules/{rule_id}	iotda:routingrules:modify
	删除规则触发条件	DELETE /v5/iot/{project_id}/routing-rule/rules/{rule_id}	iotda:routingrules:delete
流转规则动作管理	创建规则动作	POST /v5/iot/{project_id}/routing-rule/actions	iotda:routingactions:create
	查询规则动作列表	GET /v5/iot/{project_id}/routing-rule/actions	iotda:routingactions:queryList
	查询规则动作	GET /v5/iot/{project_id}/routing-rule/actions/{action_id}	iotda:routingactions:query
	修改规则动作	PUT /v5/iot/{project_id}/routing-rule/actions/{action_id}	iotda:routingactions:modify
	删除规则动作	DELETE /v5/iot/{project_id}/routing-rule/actions/{action_id}	iotda:routingactions:delete
联动规则管理	创建规则	POST /v5/iot/{project_id}/rules	iotda:rules:create
	查询规则列表	GET /v5/iot/{project_id}/rules	iotda:rules:queryList
	修改规则	PUT /v5/iot/{project_id}/rules/{rule_id}	iotda:rules:modify
	查询规则	GET /v5/iot/{project_id}/rules/{rule_id}	iotda:rules:query
	删除规则	DELETE /v5/iot/{project_id}/rules/{rule_id}	iotda:rules:delete
	修改规则状态	PUT /v5/iot/{project_id}/rules/{rule_id}/status	iotda:rules:modifyStatus
群组管理	添加设备组	POST /v5/iot/{project_id}/device-group	iotda:group:create
	查询设备组列表	GET /v5/iot/{project_id}/device-group	iotda:group:queryList
	查询设备组	GET /v5/iot/{project_id}/device-group/{group_id}	iotda:group:query
	修改设备组	PUT /v5/iot/{project_id}/device-group/{group_id}	iotda:group:modify
	删除设备组	DELETE /v5/iot/{project_id}/device-group/{group_id}	iotda:group:delete
	管理设备组中的设备	POST /v5/iot/{project_id}/device-group/{group_id}/action	iotda:group:addDevice
	查询设备组设备列表	GET /v5/iot/{project_id}/device-group/{group_id}/devices	iotda:group:queryDeviceList

设备标签管理	绑定标签	POST /v5/iot/{project_id}/tags/bind-resource	iotda:tags:bind
	解绑标签	POST /v5/iot/{project_id}/tags/unbind-resource	iotda:tags:unbind
	按标签查询资源	POST /v5/iot/{project_id}/tags/query-resources	iotda:tags:query
资源空间管理	查询资源空间列表	GET /v5/iot/{project_id}/apps	iotda:apps:queryList
	创建资源空间	POST /v5/iot/{project_id}/apps	iotda:app:create
	查询资源空间	GET /v5/iot/{project_id}/apps/{app_id}	iotda:apps:query
	删除资源空间	DELETE /v5/iot/{project_id}/apps/{app_id}	iotda:apps:delete
批量任务管理	创建批量任务	POST /v5/iot/{project_id}/batchtasks	iotda:batchtasks:create
	查询批量任务列表	GET /v5/iot/{project_id}/batchtasks	iotda:batchtasks:queryList
	查询批量任务	GET /v5/iot/{project_id}/batchtasks/{task_id}	iotda:batchtasks:query
	批量任务重试	POST /v5/iot/{project_id}/batchtasks/{task_id}/retry	iotda:batchtasks:retry
	批量任务停止	POST /v5/iot/{project_id}/batchtasks/{task_id}/stop	iotda:batchtasks:stop
	删除批量任务	DELETE /v5/iot/{project_id}/batchtasks/{task_id}	iotda:batchtasks:delete
批量任务文件管理	上传批量任务文件	POST /v5/iot/{project_id}/batchtask-files	iotda:batchtasks:stop
	查询批量任务文件列表	GET /v5/iot/{project_id}/batchtask-files	iotda:batchtaskfiles:queryList
	删除批量任务文件	DELETE /v5/iot/{project_id}/batchtask-files/{file_id}	iotda:batchtaskfiles:delete
证书管理	上传设备CA证书	POST /v5/iot/{project_id}/certificates	iotda:certificates:upload
	获取设备CA证书列表	GET /v5/iot/{project_id}/certificates	iotda:certificates:queryList
	删除设备CA证书	DELETE /v5/iot/{project_id}/certificates/{certificate_id}	iotda:certificates:delete
	验证设备CA证书	POST /v5/iot/{project_id}/certificates/{certificate_id}/action	iotda:certificates:check

软固件升级包管理	创建OTA升级包	POST /v5/iot/{project_id}/ota-upgrades/packages	iotda:otapackages:create
	查询OTA升级包列表	GET /v5/iot/{project_id}/ota-upgrades/packages	iotda:otapackages:queryList
	获取OTA升级包详情	GET /v5/iot/{project_id}/ota-upgrades/packages/{package_id}	iotda:otapackages:query
	删除OTA升级包	DELETE /v5/iot/{project_id}/ota-upgrades/packages/{package_id}	iotda:otapackages:delete
隧道管理	查询隧道列表	GET /v5/iot/{project_id}/tunnels	iotda:tunnel:queryList
	创建设备隧道	POST /v5/iot/{project_id}/tunnels	iotda:tunnel:create
	删除设备隧道	DELETE /v5/iot/{project_id}/tunnels/{id}	iotda:tunnel:delete
	查询隧道详情	GET /v5/iot/{project_id}/tunnels/{id}	iotda:tunnel:query
	修改设备隧道	PUT /v5/iot/{project_id}/tunnels/{id}	iotda:tunnel:update

1.6 应用示例

1.6.1 示例一：使用模板文件批量创建设备

场景描述

本章节指导用户通过API批量创建设备。API的调用方法参见**1.2 如何调用API**。

物联网平台支持通过请求参数和模板文件两种方式批量创建设备。本节以模板文件的方式为例，介绍如何批量创建设备。

涉及接口

- **1.4.1.2 查询产品列表**：确定待创建设备所属的产品。
- **1.4.15.7.1 上传批量任务文件**：填写批量任务文件内容并上传，确定待批量创建设备的内容。
- **1.4.15.1 创建批量任务**：通过模板文件批量创建设备。
- **1.4.15.3 查询批量任务**：确认批量创建设备结果。

操作步骤

步骤1 确定待创建设备所属的产品。

1. 查询产品列表

- 接口信息

  URL：GET /v5/iot/{project_id}/products

  详情参见**1.4.1.2 查询产品列表**

- 请求示例

```
GET https://{Endpoint}/v5/iot/{project_id}/products?
limit={limit}&marker={marker}&app_id={app_id}&offset={offset}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

- 响应示例

  Status Code: 200 OK

```
Content-Type: application/json

{
  "products" : [ {
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "app_name" : "testAPP01",
    "product_id" : "5ba24f5ebbe8f56f5a14f605",
    "name" : "Thermometer",
    "device_type" : "Thermometer",
    "protocol_type" : "CoAP",
    "data_format" : "binary",
    "manufacturer_name" : "ABC",
    "industry" : "smartCity",
    "description" : "this is a thermometer produced by Huawei",
    "create_time" : "20190303T081011Z"
  } ],
  "page" : {
    "count" : 10,
    "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

2. 根据实际需要选择产品，并记录产品的product_id。

步骤2 下载批量任务模板文件。

访问**这里**，下载模板文件。

步骤3 填写模板文件中待创建设备的参数。

参考**1.4.2.1 创建设备**，编辑**2**中下载的模板文件，填写各项参数。

样例如下：

	A	B	C	D	E	F	G	H	I	J	K	L
1	node_id	product_id	device_name	description	gateway_id	app_id	device_id	auth_type	secret	fingerprint	secure_access	timeout
2	861860000004	5ba24f5ebbe	测试设备001									
3	861860000004	5ba24f5ebbe	测试设备002									
4												

步骤4 上传批量任务文件。

1. 上传批量任务文件

- 接口信息

  URL：POST /v5/iot/{project_id}/batchtask-files

  详情参见**1.4.15.7.1 上传批量任务文件**

- 请求示例

```
POST https://{Endpoint}/v5/iot/{project_id}/batchtask-files
Content-Type: multipart/form-data
X-Auth-Token: ********
Instance-Id: ********
```

- 响应示例

  Status Code: 201 Created

```
Content-Type: application/json

{
  "file_id" : "0c3c77dd-42a2-4309-9e10-da2e8bf64ac3",
  "file_name" : "BatchCreateDevices_test01.xlsx",
  "upload_time" : "20200617T081608Z"
}
```

2. 记录批量任务文件“file_id”。

步骤5 创建批量任务

1. 创建批量任务
   - 接口信息

     URL：POST /v5/iot/{project_id}/batchtasks

     详情参见**1.4.15.1 创建批量任务**
   - 请求示例

```
POST https://{Endpoint}/v5/iot/{project_id}/batchtasks
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********

{
  "app_id" : "Ev8FVvCfOdQDzrFrxSOemiw_aMca",
  "task_name" : "BatchCommandTask",
  "task_type" : "softwareUpgrade",
  "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ],
  "targets_filter" : {
    "group_ids" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
  },
  "document" : {
    "package_id" : "32822e5744a45ede319d2c50"
  },
  "task_policy" : {
    "schedule_time" : "20151212T121212Z",
    "retry_count" : 5,
    "retry_interval" : 60
  },
  "document_source" : "jeQDJQZltU8iKgFFoW060F5SGZka"
}
```

   - 响应示例

     Status Code: 201 Created

```
Content-Type: application/json

{
  "task_id" : "5c8ba99030344005c02316ad",
  "task_name" : "testname",
  "task_type" : "softwareUpgrade",
  "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ],
  "targets_filter" : {
    "group_ids" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
  },
  "document" : {
    "package_id" : "32822e5744a45ede319d2c50"
  },
  "task_policy" : {
    "schedule_time" : "20151212T121212Z",
    "retry_count" : 5,
    "retry_interval" : 60
  },
  "status" : "Success",
  "status_desc" : "string",
```



```
 "task_progress" : {
  "total" : 0,
  "processing" : 0,
  "success" : 0,
  "fail" : 0,
  "waitting" : 0,
  "fail_wait_retry" : 0,
  "stopped" : 0
 },
 "create_time" : "20151212T121212Z"
}
```

2. 记录批量任务“task_id”。

步骤6 查询批量任务

1. 查询批量任务
   - 接口信息

     URL：GET /v5/iot/{project_id}/batchtasks/{task_id}

     详情参见**1.4.15.3 查询批量任务**
   - 请求示例

```
GET https://{Endpoint}/v5/iot/{project_id}/batchtasks/{task_id}?
limit={limit}&marker={marker}&offset={offset}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

   - 响应示例

     Status Code: 200 OK

```
Content-Type: application/json

{
 "batchtasks" : {
  "task_id" : "5c8ba99030344005c02316ad",
  "task_name" : "testname",
  "task_type" : "softwareUpgrade",
  "targets" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ],
  "targets_filter" : {
   "group_ids" : [ "e495cf17-ff79-4294-8f64-4d367919d665" ]
  },
  "document" : {
   "package_id" : "32822e5744a45ede319d2c50"
  },
  "task_policy" : {
   "schedule_time" : "20151212T121212Z",
   "retry_count" : 5,
   "retry_interval" : 60
  },
  "status" : "Success",
  "status_desc" : "string",
  "task_progress" : {
   "total" : 0,
   "processing" : 0,
   "success" : 0,
   "fail" : 0,
   "waitting" : 0,
   "fail_wait_retry" : 0,
   "stopped" : 0
  },
  "create_time" : "20151212T121212Z"
 },
 "task_details" : [ {
  "target" : "e495cf17-ff79-4294-8f64-4d367919d665",
  "status" : "Success",
  "output" : "success",
  "error" : {
```



```
    "error_code" : "IOTDA.000002",
    "error_msg" : "The request is unauthorized."
   }
  } ],
  "page" : {
   "count" : 10,
   "marker" : "5c90fa7d3c4e4405e8525079"
  }
}
```

2. 确认任务执行结果。
   根据查询结果，确认批量创建设备任务完成情况。

----结束

1.6.2 示例二：给指定设备下发消息

场景描述

本章节指导用户通过API给指定设备下发消息。API的调用方法参见**1.2 如何调用API**。

涉及接口

- **1.4.2.2 查询设备列表**：确定待下发消息的设备。
- **1.4.3.1 下发设备消息**：给指定设备下发消息。
- **1.4.3.3 查询指定消息id的消息**：确认消息下发结果。

操作步骤

步骤1 确定待下发消息的设备。

1. 查询设备列表
   - 接口信息
     URL：GET /v5/iot/{project_id}/devices
     详情参见**1.4.2.2 查询设备列表**
   - 请求示例

```
GET https://{Endpoint}/v5/iot/{project_id}/devices?
product_id={product_id}&gateway_id={gateway_id}&is_cascade_query={is_cascade_query}&node_
id={node_id}&device_name={device_name}&limit={limit}&marker={marker}&offset={offset}&star
t_time={start_time}&end_time={end_time}&app_id={app_id}
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********
```

   - 响应示例
     Status Code: 200 OK

```
Content-Type: application/json

{
 "devices" : [ {
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "description" : "watermeter device",
  "product_name" : "Thermometer",
  "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "sw_version" : "1.1.0",
  "tags" : [ {
   "tag_value" : "testTagValue",
   "tag_key" : "testTagName"
  } ],
```



```
    "app_name" : "testAPP01",
    "device_name" : "dianadevice",
    "node_type" : "ENDPOINT",
    "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
    "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
    "fw_version" : "1.1.0",
    "node_id" : "ABC123456789",
    "status" : "INACTIVE"
  } ],
  "page" : {
    "marker" : "5c8f3d2d3df1f10d803adbda",
    "count" : 100
  }
}
```

2. 根据实际需要选择设备，并记录设备的“device_id”。

步骤2 给指定**设备**下发消息。

1. 下发设备消息
   - 接口信息

     URL：POST /v5/iot/{project_id}/devices/{device_id}/messages

     详情参见**1.4.3.1 下发设备消息**
   - 请求示例

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

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

   - 响应示例

     Status Code: 201 Created

```
Content-Type: application/json

{
  "message_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "result" : {
    "status" : "PENDING",
    "created_time" : "20151212T121212Z",
    "finished_time" : "20151212T121213Z"
  }
}
```

2. 记录响应中的消息id，即“message_id”。

步骤3 确认消息下发结果。

1. 查询指定**消息id**（即message_id）的消息。
   - 接口信息

     URL：GET /v5/iot/{project_id}/devices/{device_id}/messages/{message_id}

     详情参见**1.4.3.3 查询指定消息id的消息**
   - 请求示例

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


- 响应示例

  Status Code: 200 OK

```
Content-Type: application/json

{
  "message_id" : "b1224afb-e9f0-4916-8220-b6bab568e888",
  "name" : "message_name",
  "message" : "string",
  "topic" : "string",
  "status" : "PENDING",
  "created_time" : "20151212T121212Z",
  "finished_time" : "20151212T121212Z"
}
```

2. 根据查询结果中的status字段内容确认消息下发结果。

----结束

1.6.3 示例三：在指定资源空间下创建设备

场景描述

本章节指导用户通过API在指定资源空间下创建设备。API的调用方法参见**1.2 如何调用API**。

物联网平台默认为用户创建了一个资源空间，创建设备时默认将设备归属于默认资源空间，如果用户需要对设备分资源空间管理，可以指定资源空间创建设备。

涉及接口

- **1.4.14.2 创建资源空间**：创建非系统默认的资源空间。
- **1.4.1.1 创建产品**：在指定资源空间下创建产品。
- **1.4.2.1 创建设备**：在指定资源空间下创建设备。

操作步骤

步骤1 创建非系统默认的资源空间。

1. 创建资源空间
   - 接口信息

     URL：POST /v5/iot/{project_id}/apps

     详情参见**1.4.14.2 创建资源空间**
   - 请求示例

```
POST https://{Endpoint}/v5/iot/{project_id}/apps
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********

{
  "app_name" : "testApp"
}
```

   - 响应示例

     Status Code: 201 Created

```
Content-Type: application/json

{
  "applications" : [ {
```



```
  "app_id" : "0ab87ceecbfc49acbcc8d5acdef3c68c",
  "app_name" : "testApp",
  "create_time" : "20151212T121212Z",
  "default_app" : true
 } ]
}
```

2. 记录返回结果中资源空间id，即“app_id”。

步骤2 在指定**资源空间**下创建产品。

1. 使用**这里**记录的app_id创建产品。
   - 接口信息

     URL：POST /v5/iot/{project_id}/products

     详情参见**1.4.1.1 创建产品**
   - 请求示例

```
POST https://{Endpoint}/v5/iot/{project_id}/products
Content-Type: application/json
X-Auth-Token: ********
Instance-Id: ********

{
 "product_id" : "5ba24f5ebbe8f56f5a14f605",
 "name" : "Thermometer",
 "device_type" : "Thermometer",
 "protocol_type" : "CoAP",
 "data_format" : "binary",
 "manufacturer_name" : "ABC",
 "industry" : "smartCity",
 "description" : "this is a thermometer produced by Huawei",
 "service_capabilities" : [ {
  "service_type" : "temperature",
  "service_id" : "temperature",
  "description" : "temperature",
  "properties" : [ {
   "unit" : "centigrade",
   "min" : "1",
   "method" : "R",
   "max" : "100",
   "data_type" : "decimal",
   "description" : "force",
   "step" : 0.1,
   "default_value" : {
    "color" : "red",
    "size" : 1
   },
   "enum_list" : [ "string" ],
   "required" : true,
   "property_name" : "temperature",
   "max_length" : 100
  } ],
  "commands" : [ {
   "command_name" : "reboot",
   "responses" : [ {
    "response_name" : "ACK",
    "paras" : [ {
     "unit" : "km/h",
     "min" : "1",
     "max" : "100",
     "para_name" : "force",
     "data_type" : "string",
     "description" : "force",
     "step" : 0.1,
     "enum_list" : [ "string" ],
     "required" : false,
     "max_length" : 100
    } ]
```



```
    } ],
    "paras" : [ {
      "unit" : "km/h",
      "min" : "1",
      "max" : "100",
      "para_name" : "force",
      "data_type" : "string",
      "description" : "force",
      "step" : 0.1,
      "enum_list" : [ "string" ],
      "required" : false,
      "max_length" : 100
    } ]
  } ],
  "events" : [ {
    "event_type" : "reboot",
    "paras" : [ {
      "unit" : "km/h",
      "min" : "1",
      "max" : "100",
      "para_name" : "force",
      "data_type" : "string",
      "description" : "force",
      "step" : 0.1,
      "enum_list" : [ "string" ],
      "required" : false,
      "max_length" : 100
    } ]
  } ],
  "option" : "Mandatory"
 } ],
 "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka"
}
```

– 响应示例

Status Code: 201 Created

```
Content-Type: application/json

{
  "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
  "app_name" : "testAPP01",
  "product_id" : "5ba24f5ebbe8f56f5a14f605",
  "name" : "Thermometer",
  "device_type" : "Thermometer",
  "protocol_type" : "CoAP",
  "data_format" : "binary",
  "manufacturer_name" : "ABC",
  "industry" : "smartCity",
  "description" : "this is a thermometer produced by Huawei",
  "service_capabilities" : [ {
    "service_id" : "temperature",
    "service_type" : "temperature",
    "properties" : [ {
      "property_name" : "temperature",
      "required" : true,
      "data_type" : "decimal",
      "min" : 1,
      "max" : 100,
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "centigrade",
      "method" : "R",
      "description" : "force",
      "default_value" : {
        "color" : "red",
        "size" : 1
      }
    } ],
    "commands" : [ {
```



```
    "command_name" : "reboot",
    "paras" : [ {
      "para_name" : "force",
      "required" : false,
      "data_type" : "string",
      "min" : 1,
      "max" : 100,
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "km/h",
      "description" : "force"
    } ],
    "responses" : [ {
      "paras" : [ {
        "para_name" : "force",
        "required" : false,
        "data_type" : "string",
        "min" : 1,
        "max" : 100,
        "max_length" : 100,
        "step" : 0.1,
        "unit" : "km/h",
        "description" : "force"
      } ],
      "response_name" : "ACK"
    } ]
  } ],
  "events" : [ {
    "event_type" : "reboot",
    "paras" : [ {
      "para_name" : "force",
      "required" : false,
      "data_type" : "string",
      "min" : 1,
      "max" : 100,
      "max_length" : 100,
      "step" : 0.1,
      "unit" : "km/h",
      "description" : "force"
    } ]
  } ],
  "description" : "temperature",
  "option" : "Mandatory"
 } ],
 "create_time" : "20190303T081011Z"
}
```


2. 记录返回结果中产品id，即“product_id”。

步骤3 在指定**资源空间**下创建设备。

1. 使用**这里**记录的“app_id”和**这里**记录的“product_id”创建设备。
   - 接口信息

     URL：POST /v5/iot/{project_id}/devices

     详情参见**1.4.2.1 创建设备**
   - 请求示例

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

{
  "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
  "node_id" : "ABC123456789",
  "device_name" : "dianadevice",
  "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
  "auth_info" : {
```



```
  "auth_type" : "SECRET",
  "secure_access" : true,
  "fingerprint" : "dc0f1016f495157344ac5f1296335cff725ef22f",
  "secret" : "3b935a250c50dc2c6d481d048cefdc3c",
  "timeout" : 300
 },
 "description" : "watermeter device",
 "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
 "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
 "extension_info" : {
  "aaa" : "xxx",
  "bbb" : 0
 },
 "shadow" : [ {
  "desired" : {
   "temperature" : "60"
  },
  "service_id" : "WaterMeter"
 } ]
}
```

- 响应示例

  Status Code: 201 Created

```
Content-Type: application/json

{
 "app_id" : "jeQDJQZltU8iKgFFoW060F5SGZka",
 "app_name" : "testAPP01",
 "device_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
 "node_id" : "ABC123456789",
 "gateway_id" : "d4922d8a-6c8e-4396-852c-164aefa6638f",
 "device_name" : "dianadevice",
 "node_type" : "ENDPOINT",
 "description" : "watermeter device",
 "fw_version" : "1.1.0",
 "sw_version" : "1.1.0",
 "auth_info" : {
  "auth_type" : "SECRET",
  "secure_access" : true,
  "fingerprint" : "dc0f1016f495157344ac5f1296335cff725ef22f",
  "secret" : "3b935a250c50dc2c6d481d048cefdc3c",
  "timeout" : 300
 },
 "product_id" : "b640f4c203b7910fc3cbd446ed437cbd",
 "product_name" : "Thermometer",
 "status" : "INACTIVE",
 "create_time" : "20190303T081011Z",
 "tags" : [ {
  "tag_value" : "testTagValue",
  "tag_key" : "testTagName"
 } ],
 "extension_info" : {
  "aaa" : "xxx",
  "bbb" : 0
 }
}
```

2. 根据响应确认设备创建结果。

----结束

1.7 附录

1.7.1 状态码

状态码如**表1-840**所示

表 1-840 状态码

状态码	编码	错误码说明
100	Continue	继续请求。 这个临时响应用来通知客户端，它的部分请求已经被服务器接收，且仍未被拒绝。
101	Switching Protocols	切换协议。只能切换到更高级的协议。 例如，切换到HTTP的新版本协议。
201	Created	创建类的请求完全成功。
202	Accepted	已经接受请求，但未处理完成。
203	Non-Authoritative Information	非授权信息，请求成功。
204	NoContent	请求完全成功，同时HTTP响应不包含响应体。 在响应OPTIONS方法的HTTP请求时返回此状态码。
205	Reset Content	重置内容，服务器处理成功。
206	Partial Content	服务器成功处理了部分GET请求。
300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择。
301	Moved Permanently	永久移动，请求的资源已被永久的移动到新的URI，返回信息会包括新的URI。
302	Found	资源被临时移动。
303	See Other	查看其它地址。 使用GET和POST请求查看。
304	Not Modified	所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。
305	Use Proxy	所请求的资源必须通过代理访问。
306	Unused	已经被废弃的HTTP状态码。
400	BadRequest	非法请求。 建议直接修改该请求，不要重试该请求。
401	Unauthorized	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
402	Payment Required	保留请求。

状态码	编码	错误码说明
403	Forbidden	请求被拒绝访问。 返回该状态码，表明请求能够到达服务端，且服务端能够理解用户请求，但是拒绝做更多的事情，因为该请求被设置为拒绝访问，建议直接修改该请求，不要重试该请求。
404	NotFound	所请求的资源不存在。 建议直接修改该请求，不要重试该请求。
405	MethodNotAllowed	请求中带有该资源不支持的方法。 建议直接修改该请求，不要重试该请求。
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求。
407	Proxy Authentication Required	请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权。
408	Request Time-out	服务器等候请求时发生超时。 客户端可以随时再次提交该请求而无需进行任何更改。
409	Conflict	服务器在完成请求时发生冲突。 返回该状态码，表明客户端尝试创建的资源已经存在，或者由于冲突请求的更新操作不能被完成。
410	Gone	客户端请求的资源已经不存在。 返回该状态码，表明请求的资源已被永久删除。
411	Length Required	服务器无法处理客户端发送的不带Content-Length的请求信息。
412	Precondition Failed	未满足前提条件，服务器未满足请求者在请求中设置的其中一个前提条件。
413	Request Entity Too Large	由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息。
414	Request-URI Too Large	请求的URI过长（URI通常为网址），服务器无法处理。
415	Unsupported Media Type	服务器无法处理请求附带的媒体格式。
416	Requested range not satisfiable	客户端请求的范围无效。
417	Expectation Failed	服务器无法满足Expect的请求头信息。

状态码	编码	错误码说明
422	UnprocessableEntity	请求格式正确，但是由于含有语义错误，无法响应。
429	TooManyRequests	表明请求超出了客户端访问频率的限制或者服务端接收到多于它能处理的请求。建议客户端读取相应的Retry-After首部，然后等待该首部指出的时间后再重试。
500	InternalServerError	表明服务端能被请求访问到，但是不能理解用户的请求。
501	Not Implemented	服务器不支持请求的功能，无法完成请求。
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求。
503	ServiceUnavailable	被请求的服务无效。 建议直接修改该请求，不要重试该请求。
504	ServerTimeout	请求在给定的时间内无法完成。客户端仅在为请求指定超时（Timeout）参数时会得到该响应。
505	HTTP Version not supported	服务器不支持请求的HTTP协议的版本，无法完成处理。

1.7.2 获取项目 ID

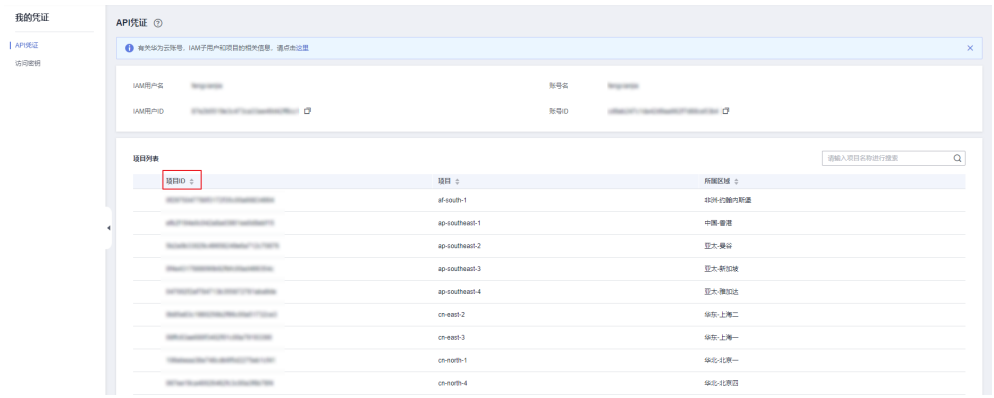
从控制台获取项目 ID

在调用接口时，部分URI以及请求体中需要填入项目ID，项目ID获取步骤如下：

1. 在华为云首页右上角，单击“控制台”。
2. 在右上角的用户名中选择“我的凭证”。

3. 在“我的凭证”界面，API凭证页签中，查看项目ID。

图 1-3 API 凭证-获取项目 ID



调用 API 获取项目 ID

项目ID还可以通过调用**查询指定条件下的项目信息**API获取。

获取项目ID的接口为“GET https://{Endpoint}/v3/projects”，其中{Endpoint}为IAM的终端节点，可以从**地区和终端节点**获取。接口的认证鉴权请参见**认证鉴权**。

响应示例如下，其中projects下的“id”即为项目ID。

```
{
    "projects": [
        {
            "domain_id": "65382450e8f64ac0870cd180d...",
            "is_domain": false,
            "parent_id": "65382450e8f64ac0870cd180d1...",
            "name": "cn-north-4",
            "description": "",
            "links": {
                "next": null,
                "previous": null,
                "self": "https://www.example.com/v3/projects/a4a5d4098fb4474fa22cd0..."
            },
            "id": "a4a5d4098fb4474fa22cd05f897xxxx",
            "enabled": true
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "https://www.example.com/v3/projects"
    }
}
```

1.7.3 错误码

当您调用API时，如果遇到“APIGW”开头的错误码，请参见**API网关错误码**进行处理。

更多服务错误码请参见**API错误中心**。

状态码	错误码	错误信息	描述	处理措施
200	IOTDA.014111	Command request timed out. Check whether the device returns a response within the specified time after receiving the request.	同步命令等待设备回复命令响应超时。	该接口为同步接口，需要设备在收到命令后回复命令响应请检查设备收到请求后是否在指定时间内返回响应给平台。
400	IOTDA.000006	Invalid input data.	请求参数不合法。	请排查请求参数是否符合华为云对应接口文档要求。
400	IOTDA.000008	Invalid input. The request format is invalid. For details, see the error message.	请求的格式不正确，如json非法，mediaType不正确等。	请排查该请求的请求格式是否正确。
400	IOTDA.000009	Invalid input. Invalid time format.	时间的格式不正确。	请排查请求参数中时间的格式是否与对应接口文档中的保持一致。
400	IOTDA.000010	Invalid input. The start time must be earlier than the end time.	请求中的开始时间大于结束时间。	请求中开始时间必须早于结束时间。
400	IOTDA.000011	Invalid input. The specified parameter 'pageNo' is out of range.	请求参数中pageNo超出范围。	请排查请求参数中的pageNo大小是否在对应接口文档限制范围之内。
400	IOTDA.000012	Invalid input. The specified parameter 'pageSize' is out of range.	请求参数中pageSize超出范围。	请排查请求参数中的pageSize大小是否在对应接口文档限制范围之内。
400	IOTDA.000013	Invalid input. The value of 'pageSize' multiplying 'pageNo' exceeds the upper limit.	查询范围超过最大限制。	请检查pageSize和pageNo参数的大小。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.000014	Invalid input. The specified parameter 'nextToken' is out of range.	请求中nextToken参数超过范围。	请排查请求中的nextToken参数是否在对应接口文档的限制范围之内。
400	IOTDA.000017	Invalid input. The specified parameter 'limit' is out of range.	请求中limit参数超过范围。	请排查请中求的limit参数是否在对应接口文档的限制范围之内。
400	IOTDA.000018	Invalid input. The specified parameter 'marker' is out of range.	请求中marker参数超过范围。	请排查请求中的marker参数是否在对应接口文档的限制范围之内。
400	IOTDA.000030	Failed to register the resource in Stage. Please try again later.	注册资源失败。	请稍后重试或联系华为工程师分析解决。
400	IOTDA.000031	Failed to deregister the resource in Stage. Please try again later.	注销资源失败。	请稍后重试或联系华为工程师分析解决。
400	IOTDA.000032	Failed to update the resource in Stage. Please try again later.	修改资源失败。	请稍后重试或联系华为工程师分析解决。
400	IOTDA.001001	Invalid input for this application.	资源空间参数不合法。	请参考华为云文档**创建资源空间**的请求参数章节。
400	IOTDA.001004	AppId is not in request header.	该用户有多个资源空间的情况下未携带appId访问接口。	请携带对应的appId。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.001011	Invalid input. The specified parameter 'app_id' is not carried.	未携带参数app_id。	请在请求参数中携带app_id。
400	IOTDA.001012	Invalid input. The appId already exists.	资源空间ID已存在。	请更换资源空间ID。
400	IOTDA.001013	Invalid input. The source and target instance IDs cannot be the same.	源实例ID与目的实例ID不能相同。	请检查实例ID是否正确。
400	IOTDA.004002	Invalid input. The tag_key %s cannot start with 'iot_'.	标签的tag_key不能以iot_开头。	请修改tag_key后进行操作。
400	IOTDA.004003	Invalid input. The tag_key %s does not exist.	不存在此tag_key的标签。	请确认传递的tag_key参数是否正确或是否存在该标签。
400	IOTDA.004004	Invalid input. This tag key %s already exists.	标签的tag_key重复。	请修改tag_key的值后再操作。
400	IOTDA.005002	Upgrade Failed. Invalid device version.	升级失败，设备版本号不合法。	请确认设备上报版本号是否为空。
400	IOTDA.005003	Upgrade failed. Verify device version failed.	升级失败， 版本号校验失败。	请将设备上报的版本号与软固件包的版本号保持一致。
400	IOTDA.005004	Upgrade failed. The current version cannot be upgraded to the target version.	升级失败，当前版本不能升级到目标版本。	请确认设备上报的版本号与软固件包支持的源版本号是否一致。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.005005	Upgrade failed. Invalid update status.	升级失败, 更新状态非法。	请确认上报的更新状态是否正确。
400	IOTDA.005006	Upgrade Failed. ErrorCode: %s, description : %s.	升级失败。	请确认上报码流是否正确，ErrorCode可参考文档**设备上报升级状态** 。
400	IOTDA.005007	Upgrade failed. The format of the device data is invalid.	升级失败， 设备上报的数据格式不合法。	请查阅**软固件升级**页面并对设备上报数据格式进行排查。
400	IOTDA.005008	Upgrade failed. Reported progress %d%% is invalid.	升级失败， 上报升级进度不在0-100之内。	请将上报升级进度约束在0-100之内。
400	IOTDA.005009	Upgrade failed. Invalid result_code.	升级失败, result_code不合法。	请查阅**设备上报升级状态** 页面“paras参数列表”部分并对参数result_code进行检查。
400	IOTDA.005010	Upgrade failed. Waiting for %s timed out.	升级失败，等待超时。	请设备及时回响应，由于设备没有及时回响应，导致任务超时失败。
400	IOTDA.005011	Upgrade failed. Send %s command failed.	升级失败，发送命令失败。	请联系华为工程师分析解决。
400	IOTDA.009002	The resource model does not exist.	资源模型不存在。	请确认请求参数中是否携带resource和event，或者notifyType。
400	IOTDA.009004	The subscription subject does not belong to the current application.	该订阅记录不属于当前应用。	请确认订阅记录和应用间的关系是否正确。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.009005	Invalid request callback URL.	请求中的callbackurl地址不合法。	请求中的callbackurl参数格式可参考文档**订阅推送相关问题** 。
400	IOTDA.009006	The subscription subject already exists.	该订阅记录已经存在。	该订阅已存在，无需再重复订阅。
400	IOTDA.009007	The request channel is invalid.	请求中channel参数不合法。	请排查请求中的channel参数是否符合对应规则动作/订阅接口文档channel取值要求。
400	IOTDA.009009	The filter is invalid.	filter不合法。	请联系华为工程师分析解决。
400	IOTDA.009010	The resource and event do not match.	resource和event不匹配。	请排查请求中resource与event参数是否符合对应规则动作/订阅接口文档中的对应关系要求。
400	IOTDA.010000	Invalid input for this rule.	规则参数不合法。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.010001	The rule name already exists.	规则名称已存在。	请换一个规则名称再重试。
400	IOTDA.010004	Invalid parameter in the rule condition.	规则条件不合法。	请求中SQL语句相关参数可参考文档**SQL语句**。
400	IOTDA.010005	Invalid parameter in the rule action.	规则动作参数不合法。	请排查请求中action参数是否符合华为云文档要求。
400	IOTDA.010006	Duplicate rule condition ID.	规则条件ID重复。	请重新命名规则条件ID后重试。
400	IOTDA.010007	Duplicate rule action ID.	规则动作ID重复。	请重新命名规则动作ID后重试。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.010008	The device in the rule condition does not exist.	规则条件中对应的设备不存在。	请检查请求参数是否正确或设备在平台是否已经存在。
400	IOTDA.010009	The device in the rule action does not exist.	规则动作中对应的设备不存在。	请检查请求参数是否正确或设备在平台是否已经存在。
400	IOTDA.010010	The device information in the rule condition does not exist.	规则条件中对应的设备信息不存在。	请检查请求参数是否正确或设备在平台是否已经存在。
400	IOTDA.010011	The device information in the rule action does not exist.	规则动作中对应的设备信息不存在。	请检查请求参数是否正确或设备在平台是否已经存在。
400	IOTDA.010012	The tag in the rule condition does not exist.	规则条件中对应的标签不存在。	请检查请求参数是否正确或标签在平台是否已经存在。
400	IOTDA.010013	Invalid rule parameter.	规则参数不合法。	请对请求参数进行校验。
400	IOTDA.010014	Invalid input. The rule action of the DEVICE_ALARM type can be created only in the condition of the DEVICE_DATA type.	DEVICE_ALARM类型的规则动作只允许创建在DEVICE_DATA类型的条件中。	请在创建的DEVICE_ALARM类型的规则动作时，将条件类型设置为DEVICE_DATA规则条件。
400	IOTDA.010015	Max rules (10) reached.	用户下规则数量超过上限。	请删除多余的规则后再注册。
400	IOTDA.010016	Invalid input. Only one DEVICE_ALARM rule can be created.	DEVICE_ALARM类型规则只允许创建一个。	DEVICE_ALARM类型规则已存在，无需重复注册。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.010017	Invalid input. The rule condition of the DEVICE_DATA type cannot contain both product_id and device_id.	在DEVICE_DATA类型的规则条件中不能同时存在product_id与device_id。	请确认DEVICE_DATA类型的规则条件中只存在device_id或product_id。
400	IOTDA.010018	Invalid input. Both device_id and product_id in the rule condition of the DEVICE_DATA type are empty.	在DEVICE_DATA类型的规则条件中product_id与device_id同时为空。	请确认DEVICE_DATA类型的规则条件中device_id或product_id仅有一个为空。
400	IOTDA.010019	The rule with the same condition already exists.	该规则条件的规则已经存在。	规则已存在， 无需重复注册。
400	IOTDA.010021	Invalid app_id.	app_id不合法。	请确认请求中的app_id是否正确。
400	IOTDA.010022	The rule has no action and cannot be enabled.	该规则没有规则动作， 无法激活。	请通过修改规则的接口为该规则添加规则动作。
400	IOTDA.010023	Duplicate channeldetail in the rule action.	规则动作中的channeldetail重复。	规则已存在，无需重复注册或者删除无用规则后重试。
400	IOTDA.010024	Invalid input. Invalid address.	地址非法。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.010025	Invalid input. Invalid username or password.	用户名或密码不合法。	请排查请求参数username与password是否符合华为云文档要求。
400	IOTDA.010026	Invalid input. The streamId or streamName is empty.	通道ID或通道名不存在。	请排查请求参数streamId与streamName是否符合华为云文档要求。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.010027	Invalid input. Failed to query the channel.	查询通道失败。	请排查请求参数是否与实际云服务产品参数一致。
400	IOTDA.010028	Invalid input. Invalid log_group_id or log_stream_id.	日志组ID或日志流ID不合法。	请排查请求参数log_group_id与log_stream_id是否符合华为云文档要求。
400	IOTDA.010029	Invalid input. Invalid func_urn.	函数的URN不合法。	请排查请求参数func_urn是否符合华为云文档要求。
400	IOTDA.010030	Invalid input. Connect to the database failed.	流转数据库连接失败。	请排查请求数据库连接参数是否符合华为云文档要求。
400	IOTDA.010031	Invalid input. The table name does not exist.	流转数据库表格不存在。	请排查请求table_name是否符合华为云文档要求。
400	IOTDA.010032	Invalid input. The suffix of the krb_file file is .conf, and the suffix of the keytab_file file is .keytab in the request.	请求krb_file文件后缀名是.conf 并且请求keytab_file文件后缀名是.keytab。	请排查请求krb_file与keytab_file是否符合华为云文档要求。
400	IOTDA.010033	Invalid input. The credential file does not exist.	Kerberos服务凭证不存在。	请排查配置凭证文件是否符合华为云文档要求。
400	IOTDA.010034	Invalid input. Connect to the cloud service failed.	连接云服务失败。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.010035	Invalid input. Max time range (24 hours) exceeded.	超出时间查询范围，最大范围是24小时。	请修改参数后重试。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.010036	Invalid input. The query time dimension is invalid.	查询时间维度无效。	请修改参数后重试。
400	IOTDA.010038	Invalid input. Connect to database failed due to invalid database info.	动作中的数据库信息不合法。	请排查流转数据库信息是否符合华为云文档要求。
400	IOTDA.010039	Invalid input. Repeated column in the action database.	动作中存在多个转发字段流转到目标数据库的同一个字段。	请排查转存配置中的转发字段是否符合华为云文档要求。
400	IOTDA.010040	Invalid input. Column in the action does not match that in the database.	动作中目标存储字段和流转数据库中的字段不匹配。	请排查流转数据库是否存在转存配置中的目标存储字段。
400	IOTDA.010042	The number of rules in a project has reached the upper limit.	规则数量已经达到上限。	请删除多余的规则后重试。
400	IOTDA.010044	Invalid input. The stack policy already exists.	已存在相同积压策略配置。	积压策略配置已存在，无需重复创建或者删除对应配置后重试。
400	IOTDA.010045	Invalid input. The %s parameter is invalid.	积压策略配置参数不合法。	请排查对应积压策略配置参数是否符合华为云控制台输入要求。
400	IOTDA.010047	Invalid input. The flow control policy is duplicated.	已存在相同流控策略配置。	流控策略配置已存在，无需重复创建或者删除对应配置后重试。
400	IOTDA.010048	Invalid input. The %s parameter is invalid.	流控策略配置参数不合法。	请排查对应流控策略配置参数是否符合华为云控制台输入要求。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.010050	Invalid input. Invalid address or the topic does not exist.	地址非法或者主题不存在。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.013000	The product does not exist or does not belong to the application.	产品不存在或产品不在指定资源空间下。	请排查请求参数是否正确或产品是否已经在指定资源空间下注册(不指定资源空间时，会在默认资源空间下查找产品)。
400	IOTDA.013001	The serviceType of the product does not exist.	产品服务类型不存在。	请排查该产品是否有服务类型，若没有，可调用修改产品接口添加服务类型。
400	IOTDA.013002	The properties of deviceServiceCapability do not exist.	产品的属性不存在。	请排查该产品是否有属性信息，若没有，可调用修改产品接口添加属性。
400	IOTDA.013003	Operation not allowed. The product is unavailable.	未知的产品类型。	请排查设备是否关联产品， 若没有，可调用修改设备信息接口添加产品ID。
400	IOTDA.013005	The productName has been used in the same application.	该资源空间下产品名已被使用。	请更换产品名重新操作。
400	IOTDA.013008	The product ID has been used in the same application.	该资源空间下productId已被使用。	请更换productId后重试。
400	IOTDA.013010	Invalid input. The content in the product service capability is duplicate. Check the content %s.	产品模型内容有重复的命名信息。	产品模型内容定义重复，请检查。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.013012	Invalid input. The size of product profile content has reached or exceeded limit.	产品模型内容大小超过了限制。	请将定义的产品模型内容限制在指定范围内。
400	IOTDA.013015	Invalid input. The number of content items %s in the product has reached the limit %s.	无效的输入，产品中的字段超过了限制。	检查产品中的字段数。
400	IOTDA.013501	Invalid input. Invalid topic_short_name. If the value of operation_type is not SUBSCRIBE, the value of topic_short_name cannot contain the number sign (#) or plus sign (+).	topic_short_name参数不合法。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.013502	Invalid input. The topic_short_name is duplicated under the same product.	相同产品下topic_short_name参数重复。	请重新填写topic_short_name后重试。
400	IOTDA.014001	Invalid input. The externalId parameter already exists.	externalId参数已经存在。	请更换externalId参数后重试。
400	IOTDA.014002	Invalid input. The type of externalId must be String.	externalId参数的类型必须是String类型。	请将externalId参数的类型改成String类型。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014008	Invalid input. Duplicated nodeId.	nodeId已被使用。	请更换nodeId后再重试。
400	IOTDA.014009	Invalid input. Duplicated deviceName.	该资源空间下deviceName已被使用。	请更换deviceName后再重试。
400	IOTDA.014010	Invalid input. The secretDevice cannot be empty when authType is SECRET.	当authType是SECRET时，secretDevice不能为空。	请修改参数后重试。
400	IOTDA.014011	Invalid input. The secretEncryptionType cannot be empty when authType is MQTT.	当authType为MQTT时，secretEncryptionType不能为空。	请修改参数后重试。
400	IOTDA.014012	Invalid input. The pskDevice cannot be empty when authType is PSK.	authType为PSK时，pskDevice不能为空。	请修改参数后重试。
400	IOTDA.014013	Invalid input. The SecureAccess (isSecure) must be true.	isSecure参数值必须为true。	请将isSecure参数设置成true。
400	IOTDA.014017	Invalid input. The serviceType does not exist.	serviceType参数不存在。	请检查serviceType参数与所属产品的serviceType是否相同。
400	IOTDA.014023	Gateway and sensor authentication types are inconsistent.	网关和子设备的认证类型不一致。	请保证网关和子设备的认证类型一致。
400	IOTDA.014025	Invalid input. Invalid serviceId.	serviceId不合法。	请排查请求参数serviceId是否符合华为云文档要求。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014027	Invalid input. The initialization cannot be empty when mode is INITIALIZATION.	mode为INITIALIZATION时initialization参数不能为空。	请联系华为工程师分析解决。
400	IOTDA.014028	Invalid input. The gateway is not online.	网关不在线。	请将网关设备接入后再重试。
400	IOTDA.014031	Invalid input. The device already exists.	设备已存在。	设备ID已存在，请更换参数后再重试，如deviceId。
400	IOTDA.014033	Invalid input. When the product protocol is CoAP, the password must be in hexadecimal format.	产品协议为CoAP时，密码必须为十六进制字符。	请将密码格式改为十六进制的字符。
400	IOTDA.014034	Invalid input. The serviceId or eventType does not match.	serviceId或eventType不匹配。	请检查请求参数serviceId与eventType是否与profile中定义的相同。
400	IOTDA.014035	Invalid input. The size of extension_info has reached or exceeded 1 KB.	extension_info字段大小超过1K。	请将extension_info参数的大小限制在1K以内。
400	IOTDA.014043	Invalid input. The gateway does not exist.	网关不存在。	请检查请求参数中的gateway_id是否已经在平台注册。
400	IOTDA.014051	Invalid input. The device does not exist or does not belong to the application.	设备不存在或者设备不在指定的资源空间下。	请排查请求参数是否有误并确认是否有在资源空间注册该设备。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014100	Invalid command status.	无效的命令状态。	请联系华为工程师解决。
400	IOTDA.014104	The device command cannot be canceled because it has been canceled, executed, or expired.	设备命令已被取消，到期或执行， 无法取消。	请联系华为工程师解决。
400	IOTDA.014105	Invalid parameter 'mode'.	mode参数不合法。	请重新确认mode参数填写是否正确，mode值仅可以为null、PASSIVE或者ACTIVE。
400	IOTDA.014107	Invalid input. Invalid parameter 'lifeCycle'.	lifeCycle参数不合法。	请联系华为工程师解决。
400	IOTDA.014108	Invalid parameter 'command_name'.	command_name参数不合法。	请检查service_id,command_name参数是否与profile中的相一致。
400	IOTDA.014110	Invalid input. The format of parameter 'commandBody' is not JSON.	commandBody参数格式不是json格式。	请确认请求中对应的参数paras是json格式。
400	IOTDA.014112	Send to device failed because the device does not subscribe to the topic.	发送到设备失败，设备没有订阅该主题。	请确认设备是否订阅正确的topic。
400	IOTDA.014113	Invalid input. The size of paras has exceeded the upper limit.	paras参数大小超过最大值。	请减小请求中paras参数长度。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014114	The API does not support MQTT devices.	此接口不支持MQTT协议类型的设备。	请更换为NB设备或使用支持MQTT协议的接口。
400	IOTDA.014115	The API does not support NB-IoT devices.	此接口暂不支持NB-IoT设备。	对于NB-IoT设备，请使用下发异步设备命令接口。
400	IOTDA.014116	Invalid input. The size of request body has exceeded the upper limit.	下发的请求体大小超过最大值。	请减小下发的请求体大小。
400	IOTDA.014130	Invalid input. The messageId of the device is not unique.	请求下发给设备的message_id不唯一，与之前创建的message_id重复。	message_id需要确保唯一，请修改message_id参数。
400	IOTDA.014150	Invalid input. The topic has no permission.	非法的输入，该topic没有订阅的权限。	请检查topic参数，确保topic输入正确且该topic在产品中具备订阅权限或者全部权限。
400	IOTDA.014151	Invalid input. The topic-related parameters in the request are duplicate.	非法的输入，topic相关的参数重复。	请确保请求体中topic字段与topic_full_name字段只有一个被输入。
400	IOTDA.014201	Invalid input. The batch task name already exists.	该任务名已存在。	请更换任务名后重试。
400	IOTDA.014203	Invalid input. The document parameter is invalid. errorMsg : %s.	文档中的参数不合法。	请查阅**创建批量任务** 页面“请求参数”部分并对参数document进行检查。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014204	Invalid input. The targets and targets_filter cannot both be empty.	targets和targets_filter参数不能全部为空。	请保证其中一个参数不为空。
400	IOTDA.014205	Invalid input. The key of targets_filter only supports %s.	targets_filter参数的key只支持指定类型。	请查阅**创建批量任务**页面“请求参数”部分并对参数targets_filter进行检查。
400	IOTDA.014207	Invalid input. The start time cannot be earlier than the current time, and the latest start time cannot exceed %s days.	开始时间不能早于当前时间，最晚启动事件不能超过指定天数。	请查阅**创建批量任务**页面“请求参数”部分并对参数schedule_time进行检查。
400	IOTDA.014208	Invalid input. retry_count and retry_interval depend on each other and must be assigned at the same time.	retry_count和retry_interval参数互相依赖，并且必须同时分配。	请保证retry_count和retry_interval参数同时不为空。
400	IOTDA.014209	Invalid input. The task cannot be stopped because it is completed or being stopped.	任务不能停止，因为任务已经完成或者正在停止中。	请确认任务是否处于已完成或停止状态。
400	IOTDA.014210	Invalid input. The parameter 'targets_filter' does not support multiple keys.	targets_filter参数只支持一个key。	请排查请求参数是否符合华为云文档要求。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014216	Invalid input. The task cannot retry because the task has succeeded, stopped, being stopped or is waiting.	任务不能重试，因为任务已经成功、停止、正在停止或者等待中。	请确认任务是否处于成功、停止、正在停止或者等待中。
400	IOTDA.014219	Invalid input. The target is not in the task.	目标不在该任务中。	请确认该目标是否在该任务中。
400	IOTDA.014300	Operation not allowed. The number of certificates has reached the upper limit (%s).	证书数量达到上限。	证书数量已达到上限，请删除无用证书后重试，单租户下证书上限：100。
400	IOTDA.014301	Invalid certificate content.	证书内容不合法。	请检查证书内容是否符合X509格式要求，证书内容解析参考**证书内容解析**。
400	IOTDA.014302	Invalid input. The certificate (certificate_id: %s) already exists.	证书已存在。	您已上传相同内容证书，无需重复上传。
400	IOTDA.014303	Operation not allowed. Upload certificate failed.	证书上传失败。	请联系华为工程师分析解决。
400	IOTDA.014304	Operation not allowed. Delete certificate failed.	证书删除失败。	请联系华为工程师分析解决。
400	IOTDA.014305	Operation not allowed. Query certificate failed.	证书查询失败。	请联系华为工程师分析解决。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014307	Operation not allowed. The certificate failed to verify the verifyCode.	证书验证码校验失败。	校验证书的cn_name与当前CA证书的校验码不一致，请核对后重新生成校验证书认证。
400	IOTDA.014308	Operation not allowed. The certificate failed to verify the path.	校验证书路径失败。	校验证书路径与当前CA证书路径不一致，请确保两者是由同一个CA机构颁发。
400	IOTDA.014309	The certificate is expired or about to expire. The expiry date must be seven days later than current date.	证书即将过期或已过期。	请重新申请有效证书。
400	IOTDA.014310	The size of the certificate file cannot be larger than 1 MB and the file name must match the pattern ^[.a-zA-Z0-9_-]{1,255}$.	证书文件不能大于1MB且文件名必须符合^[.a-zA-Z0-9_-]{1,255}$正则。	请重新上传符合条件的证书。
400	IOTDA.014311	The certificate is in use and cannot be deleted.	该证书正在使用中无法删除。	请解除证书关联后再删除。
400	IOTDA.014312	Invalid input. The certificate scene does not exist.	证书场景不存在。	请确认请求参数是否正确。
400	IOTDA.014313	Invalid input. The certificate scene already exists.	证书场景已存在。	请确认请求参数是否正确。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014314	Invalid input. The server certificate does not exist.	服务器证书不存在。	请确认请求参数是否正确。
400	IOTDA.014315	Invalid input. The CA certificate does not exist.	CA证书不存在。	请确认请求参数是否正确。
400	IOTDA.014316	Invalid input. Failed to parse the private key and certificate.	私钥与证书解析失败。	请检查私钥与证书是否合法是否匹配。
400	IOTDA.014318	Invalid input. The CA certificate not bind provisioning-template.	CA证书未绑定模板。	请绑定模板后重试。
400	IOTDA.014320	Invalid input. The provisioning-template is not exist.	指定模板不存在。	请确认指定模板是否存在后重试。
400	IOTDA.014322	Invalid input. The same CA certificate has been bound provisioning-template in another application.	其他应用下已有相同CA证书绑定了模板。	请解绑其他应用下的证书模板后重试。
400	IOTDA.014602	Invalid input. The batch task file name already exists.	批量任务文件名已存在。	请更换文件名后重试。
400	IOTDA.014603	Invalid input. The size of the batch task file exceeds the upper limit.	批量任务文件的大小超过最大限制。	单个批量任务文件的大小最大限制为4M。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.014604	Invalid input. The number of lines in the batch task file exceeds the upper limit.	批量任务文件行数超过最大限制。	单个批量文件行数最大限制为30000行。
400	IOTDA.014605	Invalid input. The resource-suffix of the batch task file is wrong.	批量任务文件后缀名错误。	请查阅**上传批量任务文件**页面“请求参数”部分并对参数file进行检查。
400	IOTDA.014606	Invalid input. Invalid batch task file name.	批量任务文件名不合法。	请查阅**上传批量任务文件**页面“请求参数”部分并对参数file进行检查。
400	IOTDA.014608	Invalid input. The content of batch task file is invalid.	批量任务文件内容不合法。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.014900	Invalid input. The search SQL contains unknown field.%s.	搜索SQL语句包含不识别的字段名。	请检查搜索SQL语句，修正不合法的字段，然后重试。
400	IOTDA.014901	Invalid input.The SQL statement contains invalid parameters.%s.	搜索SQL语句包含不合法的参数，或者数据类型与运算符不匹配。	请检查搜索SQL语句，修正不合法的参数，然后重试。
400	IOTDA.014902	Invalid input.The search SQL is invalid.%s.	搜索SQL语法错误。	请检查搜索SQL语句，参考文档使用正确的语法，然后重试。
400	IOTDA.015100	Invalid input. proxy_name is already exists.	设备代理名已存在。	请更换参数proxy_name后再重试。
400	IOTDA.015200	Invalid input. policy_name is already exists.	设备策略名已存在。	请更换参数policy_name后再重试。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.015203	Invalid input. The bind target not exist or does not belong to the application.	绑定对象不存在或者绑定对象与设备策略不在同一资源空间下。	请排查请求参数是否有误并确认目标对象是否在对应资源空间下。
400	IOTDA.015205	Invalid input. The target has been bound to the device-policy.	目标对象已经绑定该设备策略。	目标对象已经绑定该设备策略，无需再绑定。
400	IOTDA.015206	Invalid input. The device-policy id does not exist.	设备策略id不存在。	设备策略id不存在，请检查参数是否有误。
400	IOTDA.015300	Invalid input. template_name is already exists.	模板名已存在。	请更换参数template_name后再重试。
400	IOTDA.015304	Invalid input. The parameter defined in the template does not exist.	证书信息不存在模板引用参数	请检查设备证书信息是否齐全。
400	IOTDA.015305	Invalid input. The field [template_body.resources.device.product_id] does not exist.	指定的产品id不存在。	请排查请求参数是否有误并确认产品id是否存在。
400	IOTDA.016000	Invalid input, The queue name already exist in the same spUserName.	队列名称在该用户下已存在。	请更换队列名称后重试。
400	IOTDA.016004	Invalid queue name.	队列名称不合法。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.016005	Invalid AMQP access configuration.	amqp对接配置不合法。	请排查请求参数是否符合华为云文档要求。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.016006	Invalid HTTP access configuration.	http对接配置不合法。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.016008	The integration configuration already exists.	对接配置已存在。	请排查请求参数是否符合华为云文档要求。
400	IOTDA.019304	Invalid input. Invalid obs info.	obs信息无效。	请确认请求参数region_name、bucket_name、object_key是否正确。
400	IOTDA.019305	Invalid input. The format of the upgrade package is incorrect.	升级包格式不正确。	请参考华为云文档**软固件包上传**中升级包格式限制。
400	IOTDA.019306	Invalid input. The upgrade package size exceeds the limit.	升级包大小超过限制。	请参考华为云文档**软固件包上传**中升级包大小限制。
400	IOTDA.019600	Invalid input. The bridge name is duplicated.	网桥名称重复。	请确认是否已经创建了相同名称的网桥。
400	IOTDA.019603	Invalid input. The bridge ID is duplicated.	网桥ID重复。	请确认是否已经创建了相同ID的网桥。
400	IOTDA.019702	The tunnel has already closed.	隧道已关闭。	隧道已关闭，请重新创建隧道。
400	IOTDA.019708	Failed to deliver tunnel information to the device.	下发隧道配置通知给设备失败。	请确认设备运行正常且订阅了接收隧道通知的主题。
400	IOTDA.021304	The number of device log configurations exceeds the upper limit.	设备日志配置数量超过用户最大配额。	请关闭部分设备设备日志收集，然后再重新开启。

状态码	错误码	错误信息	描述	处理措施
400	IOTDA.021310	Invalid input. The authorizer name is duplicated.	自定义鉴权名称重复。	请确认是否已经创建了相同名称的自定义鉴权器。
401	IOTDA.000002	Authentication failed.	鉴权失败。	请排查请求中的鉴权参数是否携带正确。
401	IOTDA.000025	SP user authentication failed.	SP Token鉴权失败。	请检查sp token是否正确。
401	IOTDA.000026	Stage user authentication failed.	stage token鉴权失败。	请检查stage token是否正确。
401	IOTDA.001003	Incorrect AppId or secret.	资源空间与密钥不匹配。	请排查该密钥是否正确。
401	IOTDA.014032	Invalid input. The time does not match.	时间不匹配。	请联系华为工程师解决。
401	IOTDA.019704	Invalid tunnel access token.	tunnel_access_token不合法。	请确认请求参数是否正确。
401	IOTDA.019705	The tunnel access token is expired.	tunnel_access_token已过期。	请重新创建隧道后重试。
403	IOTDA.000004	Invalid access token.	非法token。	请排查请求中的token是否正常。
403	IOTDA.000005	Refresh access token failed.	刷新token失败。	请排查请求中的refreshToken是否正确。
403	IOTDA.000015	The account is frozen.	账户已被冻结。	联系账户负责人进行解冻。
403	IOTDA.000021	Operation not allowed. User not found by IAM token or the authorized user has not subscribed to IoTDA.	没有找到IAM Token所对应的用户信息或该用户没有订阅设备接入服务（IOTDA）。	请排查IAM Token所在用户是否订阅了设备接入服务(IOTDA)。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.000022	Operation not allowed. The user does not have the permission.	该用户没有权限。	请排查该用户是否有权限访问。
403	IOTDA.000023	Request frequency reached the upper limit %s.	请求已经达到限制速率。	请降低请求频率，该请求速率已达到限制。
403	IOTDA.000024	Operation not allowed. Only one token is allowed in the request header.	请求头只能放置一个Token域。	请删除多余的token头域。
403	IOTDA.000028	System is being maintained. The configuration cannot be modified.	系统正在维护，请稍后重试。	请稍后重试。
403	IOTDA.000033	Operation not allowed. The current instance does not support the parameter.	当前实例规格不支持此功能参数。	请排查请求参数是否符合华为云文档要求。
403	IOTDA.000034	Operation not allowed. The user does not have the permission.	该用户没有权限。	请排查该用户是否有权限访问。
403	IOTDA.001000	The application does not exist.	该资源空间不存在。	请确定是否已在平台创建资源空间并检查资源空间ID是否正确。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.001002	Operation not allowed. You do not have the permissions required to access the application.	该应用没有权限访问。	请检查该应用是否已被授权。
403	IOTDA.001005	Operation not allowed. The parameter 'app_id' is not carried, and the authorized user has more than one application. Include the parameter 'app_id', or contact Huawei technical support engineers to merge application data.	该用户下有多个应用的情况下未携带appId访问接口。	请携带对应的appId或联系华为工程师合并应用数据。
403	IOTDA.001006	Operation not allowed. Application not found by authorized user or the authorized user has no application.	用户下没有资源空间或资源空间与用户不匹配。	请排查用户下是否有资源空间或是否有指定的资源空间。
403	IOTDA.001007	Operation not allowed. The application does not belong to the authorized user.	资源空间与用户信息不匹配。	请排查该用户下是否有指定的资源空间。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.001008	Operation not allowed. The app name already exists.	资源空间名称已存在。	请更换资源空间名称。
403	IOTDA.001009	Operation not allowed. The maximum number of applications has been reached.	资源空间数量已达到上限。	该用户资源空间数量已达到上限，请删掉多余的资源空间后再注册。
403	IOTDA.001010	Operation not allowed. The default app cannot be deleted.	禁止删除默认资源空间。	请修改参数后重试。
403	IOTDA.001014	Operation not allowed. The application does not belong to the authorized instance.	资源空间与实例信息不匹配。	请排查该实例下是否有指定的资源空间。
403	IOTDA.003002	Operation not allowed. The group name already exists.	设备组名称已被使用。	请更换设备组名称后重试，该设备组名称已被使用。
403	IOTDA.003003	Operation not allowed. Max groups (1,000) reached.	设备组数量已达到最大上限1000个。	请删除多余设备组后重试，该设备组数量已达上限。
403	IOTDA.003004	Operation not allowed. Max group depths (5) reached.	该设备组深度已达上限（5），不允许再次注册子设备组。	请更换深度较小的父设备组的ID进行注册，该群组深度已达上限，不允许再注册子群组。
403	IOTDA.003005	Operation not allowed. The device already exists in the group.	该设备在设备组中已存在。	该设备在设备组中已存在，无需再次添加。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.003006	Operation not allowed. The device does not exist in the group.	该设备在设备组中不存在。	该设备在设备组中不存在，无需删除。
403	IOTDA.003007	Operation not allowed. Max devices (20,000) reached for the group.	设备组中的设备数量达到最大限制20000。	请删除多余设备或更换群组。
403	IOTDA.003008	Operation not allowed. A device can be added to up to 10 groups.	一个设备最多只能添加到10个设备组中。	请将该设备从多余的群组中删除后重试，该设备所在群组已达到最大限制。
403	IOTDA.003009	Operation not allowed. The parent group contains child groups. Delete the child groups and try again.	该设备组为其他设备组的父设备组，若想删除此群组，需先删除其子设备组。	如需删除该设备组，请先删除该群组下的所有子设备组后再执行该操作。
403	IOTDA.003010	Operation not allowed. The group contains devices and cannot be deleted.	该群组下存在设备，禁止删除。	如需删除，请先确保该设备组下的设备无用，删除该设备组下的所有设备后再删除该设备组。
403	IOTDA.003011	Operation not allowed. The group and device do not belong to the same application.	群组和设备不属于同一个资源空间。	请使用相同资源空间下的群组和设备进行操作。
403	IOTDA.004001	Operation not allowed. Max bindable tags (10) reached for the device.	设备绑定的标签数量超过最大限制10。	请删除多余的标签再进行绑定，该设备已绑定标签数达到最大限制。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.005000	Operation not allowed. Only one task can be started for a device at a time.	一个设备同时只能开启一个任务。	请手动操作任务结束或者等待当前任务完成后再开启另一个任务。
403	IOTDA.005001	Operation not allowed. The protocol of the device does not support upgrade.	该设备的协议类型不支持升级。	请提交工单 联系华为工程师分析解决。
403	IOTDA.009001	Max application subscription records reached.	应用订阅记录数达到上限。	请删除多余订阅记录，该应用订阅数量已达到上限。
403	IOTDA.009008	The maximum number of queries has been reached.	查询数量超过最大限制。	请排查请求参数是否符合华为云文档要求。
403	IOTDA.010003	The number of rules has reached the upper limit.	规则数量已达到上限。	请删除多余的规则后重试。
403	IOTDA.010037	The rule cannot be deleted because actions exist in the rule.	规则中已存在动作，不能删除规则。	请先删除规则动作，再删除规则。
403	IOTDA.010049	Operation not allowed. Total number of flow control policies exceeds the upper limit (4).	流控策略配置数量超过最大限制4。	请删除多余的流控策略配置后重新创建。
403	IOTDA.013004	Operation not allowed. You have no write permission.	您没有可写的权限。	请排查您的产品属性是否是可写的，若不是，可调用修改产品的接口将属性改成可写。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.013006	The number of products in the application has reached the upper limit.	该资源空间下产品数量达到上限（1000）。	请删除无用的产品后重试。
403	IOTDA.013007	Operation not allowed. The product is in use and cannot be deleted.	产品已被使用，禁止删除。	删除产品需先删除该产品下设备，如需删除，请确保该产品没有关联设备，再重试删除操作。
403	IOTDA.013009	Operation not allowed. The value of default_value must be writable.	default_value必须属于可写属性。	请通过修改产品的接口将该属性设置成可写。
403	IOTDA.013011	Operation not allowed. The number of assets properties has reached the upper limit (%s).	产品资产属性定义个数超过上限。	请删除无用的资产属性定义。
403	IOTDA.013013	Operation not allowed. The protocol type cannot be changed to CoAP.	产品协议类型不支持修改为CoAP。	请重新创建CoAP协议类型产品。
403	IOTDA.013503	Operation not allowed. The number of topics in a product exceeds the upper limit (%s).	一个产品下的主题数量超过最大限制。	请删除无用的主题后再重试。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.014003	Operation not allowed. The number of frozen devices has reached the upper limit.	冻结设备的数量已达到上限。	请将非必须冻结的设备解冻后再次冻结该设备或联系华为工程师处理。
403	IOTDA.014004	Operation not allowed. This device is online.	该操作只允许设备在“离线”状态下执行。	请使设备调整为“离线”状态，再重试操作。
403	IOTDA.014005	Operation not allowed. This device is not active.	该操作只允许在设备激活后执行。	请先将设备激活。
403	IOTDA.014006	Operation not allowed. The secret cannot be reset.	密码无法被重置。	NB设备未使用密钥，无需重置密钥。
403	IOTDA.014007	Operation not allowed. The PSK cannot be reset.	PSK无法被重置。	请修改参数后重试。
403	IOTDA.014015	Operation not allowed. The number of concurrent location service requests has exceeded the upper limit.	位置服务请求的并发超过上限。	请降低接口调用频率。
403	IOTDA.014016	Operation not allowed. The device is not online.	该操作只允许设备在线后执行。	请将设备接入后再重试。
403	IOTDA.014018	Operation not allowed. The device has been frozen and cannot be operated.	该设备是冻结状态，不允许操作。	请将**设备解冻**后再重试。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.014019	Operation not allowed. The number of devices of the user has reached the upper limit.	用户下的设备数量达到上限。	请删除无用设备或联系华为工程师分析解决。
403	IOTDA.014020	Operation not allowed. The number of devices bound to this application has reached the upper limit.	资源空间下的设备数量超过最大上限。	请删除无用设备或联系华为工程师分析解决。
403	IOTDA.014021	Operation not allowed. The number of devices has reached the upper limit of the package.	套餐下的设备数量超过最大上限。	请删除无用设备或联系华为工程师分析解决。
403	IOTDA.014022	Operation not allowed. The number of license resources has reached the upper limit.	许可证资源数量已达到上限。	请联系华为工程师分析解决。
403	IOTDA.014024	Operation not allowed. The device already has an endpoint.	该网关已关联子设备，无法直接删除。	删除网关前，必须删除其关联的子设备，如需删除，请确保子设备无用或关联至其他网关，再重试网关删除操作。
403	IOTDA.014026	Operation not allowed. The number of non-security devices has reached the upper limit.	非安全设备的数量已达到上限。	请删除无用的非安全设备或联系华为工程师分析解决。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.014029	Operation not allowed. The timeout parameter cannot be modified when the device has been activated.	设备激活后不能修改timeout参数。	取消timeout参数的修改或在未激活的设备下操作。
403	IOTDA.014036	Operation not allowed. The device service capabilities are not defined in the product.	该设备的产品未定义设备服务能力。	请使用修改产品接口将该产品添加设备服务能力。
403	IOTDA.014037	Operation not allowed. Max device depths (2 levels) reached.	该设备深度已达到上限（2）。	当前网关最多支持二级子设备。
403	IOTDA.014038	Operation not allowed. Only gateways or directly connected devices are supported.	该操作只支持网关或者直连设备。	请选择直连设备或者网关设备进行操作。
403	IOTDA.014039	Operation not allowed. The device has already been frozen.	该设备是冻结状态，无法操作。	请解冻设备后再进行操作。
403	IOTDA.014040	Operation not allowed. The device is not frozen and cannot be unfrozen.	该设备不是冻结状态，无法解冻。	只能对冻结状态的设备进行解冻操作。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.014041	Operation not allowed. The device does not belong to the currently connected gateway.	该设备不属于当前连接的网关。	请选择当前连接网关下的子设备进行操作。
403	IOTDA.014101	The number of commands reached the upper limit.	该资源空间下的缓存命令数量达到上限。	请等待缓存命令下发后再下发新命令，或增大缓存命令数。
403	IOTDA.014106	Invalid CommandBody for the MQTT protocol.	适用于mqtt协议的CommandBody不正确。	命令下发接口参数必须为json格式，请确认下发的参数是否正确。
403	IOTDA.014109	Operation not allowed. The command status is not 'PENDING'.	不允许操作，命令的状态不是PENDING。	请联系华为工程师解决。
403	IOTDA.014133	Operation not allowed. The topic has no 'SUBSCRIBE' permission.	操作不允许，该主题没有SUBSCRIBE权限。	请联系华为工程师解决。
403	IOTDA.014202	Operation not allowed. The number of unfinished tasks has reached the upper limit (%s).	未完成的任务数量达到上限。	执行中的批量任务数量已经超出规格限制，请停止未完成任务再重试。
403	IOTDA.014206	Operation not allowed. The number of targets has reached the upper limit (%s).	targets数量超过上限。	请排查请求参数是否符合华为云文档要求。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.014217	Operation not allowed. The target cannot retry because the task of target has succeeded, is waiting or is processing.	目标不能重试，因为目标的任务已经成功、等待中或者正在执行中。	请确认该目标任务是否已经成功、等待中或者正在执行中。
403	IOTDA.014218	Operation not allowed. The target cannot stop because the task of target has succeeded, failed or stopped.	目标不能停止，因为目标的任务已经成功，失败或者停止。	请确认该目标任务是否已经成功，失败或者停止。
403	IOTDA.014220	Operation not allowed. The task type is not supported.	不支持该任务类型。	请确认任务类型是否支持该操作。
403	IOTDA.014221	Invalid input. The task cannot be deleted because it is not completed.	任务不能删除，因为任务没有完成。	请检查任务状态是否为成功，失败，部分成功或已停止。
403	IOTDA.014319	Operation not allowed. The CA certificate state is not verified, please verify the certificate first.	CA证书状态是未验证，请先校验CA证书。	请验证CA证书后重试。
403	IOTDA.014601	Operation not allowed. The number of batch task files of the current user exceeds the upper limit.	当前用户下的批量任务文件数达到上限。	请删除多余文件后重试。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.014607	Operation not allowed. The batch task file is in use and cannot be deleted.	该批量任务文件正在使用，不能被删除。	请等待任务完成后再操作。
403	IOTDA.015101	Operation not allowed. The device already exists in the other device-proxy.	设备已存在于其他设备代理中，不允许添加到当前设备代理。	请更换代理设备后重试。
403	IOTDA.015102	Operation not allowed. The endpoint device is not allowed to be added to device-proxy.	子设备不允许添加至设备代理中。	子设备不允许添加至代理中，请更换设备后重试。
403	IOTDA.015104	Operation not allowed. The number of device-proxy has reached the upper limit (10).	设备代理数量已达到最大上限。	请删除多余代理设备后重试。
403	IOTDA.015201	Operation not allowed. The number of device-policy has reached the upper limit (%s).	设备策略数量已经达到上限。	请删除多余的设备策略后再重试。
403	IOTDA.015204	Operation not allowed. The number of policies bound to the target reaches the upper limit.	目标对象绑定的设备策略数量达到上限。	请为目标对象解绑无用的设备策略后重试。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.015303	Operation not allowed. The provisioning-template is associated by certificate.	模板被证书关联中，不允许删除。	请在设备CA证书中解除关联后重试。
403	IOTDA.015306	Operation not allowed. The number of provisioning-template has reached the upper limit (10).	模板数量达到上限。	请删除不需要的模板后重试。
403	IOTDA.016001	Operation not allowed, The number of queues exceeds the limit for each spUserName.	队列数量已达到用户最大配额。	单个租户只能创建100个AMQP队列。
403	IOTDA.016003	Operation not allowed. The queue is in use.	该队列已被使用。	请取消该队列相应的订阅后再进行操作。
403	IOTDA.016009	Operation not allowed. The number of queues exceeds the system limit.	队列数量已达到系统最大配额。	请联系华为工程师分析解决。
403	IOTDA.019303	Operation not allowed. The number of packages for the user has reached the upper limit.	用户下的升级包数量达到上限。	请删除无用升级包。
403	IOTDA.019601	Operation not allowed. The number of bridges in the system has reached the upper limit.	租户下的网桥数已经达到最大值。	请确认租户下的网桥数量是否已经为20个。

状态码	错误码	错误信息	描述	处理措施
403	IOTDA.019701	Operation not allowed. The device does not belong to the authorized user.	设备不属于该用户。	设备不属于该用户，请确认设备ID是否正确。
403	IOTDA.019703	The open tunnel cannot be deleted.	处于打开状态的隧道不能被删除。	请先关闭设备隧道后再进行删除。
403	IOTDA.019706	Operation not allowed. The number of tunnels for a device has reached the upper limit.	单设备的隧道个数已达到限制值。	请删除老的设备隧道后再进行重试。
403	IOTDA.019707	Operation not allowed. The number of device tunnels for a user has reached the upper limit.	租户的设备隧道个数已达到限制值。	请删除老的设备隧道后再进行重试。
403	IOTDA.021311	Operation not allowed. The number of authorizer has reached the upper limit(10).	租户下的自定义鉴权数已经达到最大值。	请确认租户下的自定义鉴权数是否已经为10个。
403	IOTDA.021313	The function does not exist.	该函数在FunctionGraph不存在。	请确认该函数是否在FunctionGraph是否已经创建。
404	IOTDA.000016	Target not found.	未找到目标。	请联系华为工程师分析解决。
404	IOTDA.000029	Invalid input. The path does not exist.	请求路径不存在。	请检查请求路径是否正确。
404	IOTDA.003000	The group does not exist.	群组不存在。	请确认是否有该设备组或设备组参数是否携带正确。

状态码	错误码	错误信息	描述	处理措施
404	IOTDA.003001	The parent group does not exist.	父级设备组不存在。	请确认是否有该设备组或设备组参数是否携带正确。
404	IOTDA.004000	The resource does not exist.	资源不存在。	请确认请求参数是否正确。如deviceId，resourceId等。
404	IOTDA.009003	The subscription subject cannot be found.	查询不到该订阅记录。	请确认请求中的参数是否与待操作的订阅匹配。
404	IOTDA.010002	The rule does not exist.	该规则不存在。	请确认平台是否存在该规则或请求参数是否正确。
404	IOTDA.010020	The rule action does not exist.	规则动作不存在。	请确认平台是否存在该规则动作或请求参数是否正确。
404	IOTDA.010043	The stack policy does not exist.	积压策略配置不存在。	请确认平台是否存在对应积压策略配置或请求参数是否正确。
404	IOTDA.010046	The flow control policy does not exist.	流控策略配置不存在。	请确认平台是否存在对应流控策略配置或请求参数是否正确。
404	IOTDA.013014	The product does not exist.	产品不存在。	请排查请求参数是否正确或产品是否已经在平台注册。
404	IOTDA.013500	The topic does not exist.	该主题不存在。	请联系华为工程师解。
404	IOTDA.014000	The device does not exist.	设备不存在。	请排查请求参数是否有误并确认是否有在平台注册该设备。
404	IOTDA.014014	The IMSI does not exist.	IMSI参数值不能为空。	请输入非空IMSI值后重试。
404	IOTDA.014042	The device access information does not exist.	设备接入信息不存在。	请检查设备是否接入平台并激活。

状态码	错误码	错误信息	描述	处理措施
404	IOTDA.014103	The device command does not exist.	设备命令不存在。	请确认请求参数是否正确。
404	IOTDA.014131	The message does not exist.	该消息不存在。	请联系华为工程师解决。
404	IOTDA.014200	The batch task does not exist.	该任务不存在。	请排查请求参数中的task_id是否有效。
404	IOTDA.014306	The certificate does not exist.	证书不存在。	目标证书id不存在，请确认请求参数是否正确。
404	IOTDA.014600	The batch task file does not exist.	批量任务不存在。	请确认请求参数是否正确。
404	IOTDA.015105	device-proxy not exist or does not belong to the application.	设备代理不存在或者代理不在指定的资源空间下。	请排查请求参数是否有误并确认是否有在资源空间注册该设备代理。
404	IOTDA.015202	Invalid input. device-policy not exist or does not belong to the application.	设备策略不存在或者设备策略不在指定的资源空间下。	请排查请求参数是否有误并确认是否有在资源空间创建该设备策略。
404	IOTDA.015301	provisioning-template does not exist.	指定模板不存在。	请排查请求参数是否有误并确认资源是否存在。
404	IOTDA.016002	The queue ID does not exist.	队列ID不存在。	请确认请求参数是否正确。
404	IOTDA.019300	The package does not exist.	升级包不存在。	请确认是否有该升级包或请求参数package_id是否携带正确。
404	IOTDA.019602	Invalid input. The bridge does not exist.	指定网桥不存在。	请确认该网桥是否已经创建。

状态码	错误码	错误信息	描述	处理措施
404	IOTDA.019700	Invalid input. The tunnel does not exist.	隧道不存在。	请确认是否有隧道创建成功并且隧道ID填写正确。
404	IOTDA.021312	The authorizer does not exist.	该自定义鉴权不存在。	请确认该自定义鉴权是否已经创建。
405	IOTDA.000003	The request method is not supported.	不支持该请求方式。	请排查请求方式是否与文档中的保持一致。
405	IOTDA.000019	Method not allowed.	http请求中的请求方法不正确。	请排查请求方式是否与文档中的保持一致。
406	IOTDA.000037	Invalid input. The Accept is missing or not supported in the request header.	请求头中缺少或携带了不支持的Accept。	请排查请求头中是否携带了正确的Accept。
409	IOTDA.014030	The version of the serviceId %s conflicts with another one.	该serviceId的版本号冲突。	请使用正确的版本号进行配置。
409	IOTDA.014047	Device status update conflicts occur.	设备状态更新冲突。	请检查是否重复发送请求或网关设备已下线。
415	IOTDA.000036	Invalid input. The Content-Type is missing or not supported in the request header.	请求头中缺少或携带了不支持的Content-Type。	请排查请求头中是否携带了正确的Content-Type。
429	IOTDA.014102	Congestion occurs. The network is under flow control.	发生拥堵， 并且当前网络受到流量控制。	请联系华为工程师解决。
500	IOTDA.000001	Internal server error.	服务器内部错误。	请联系华为工程师分析解决。

状态码	错误码	错误信息	描述	处理措施
500	IOTDA.000007	The data in database is abnormal.	数据库中的数据不正常。	请联系华为工程师分析解决。
500	IOTDA.000020	Decrypt IAM token failed.	IAM Token解析失败。	请联系华为工程师分析解决。

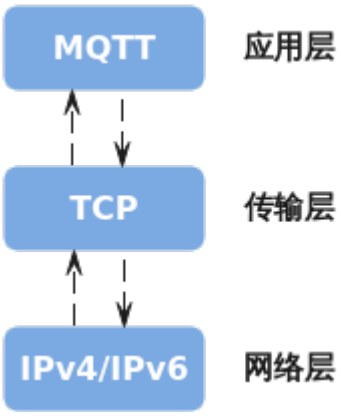
2 设备侧 MQTT/MQTTS 接口参考

2.1 使用前必读

概述

MQTT是一种底层传输协议，一般用于网络通信，它提供有序有的、可靠的、双向字节流传输，支持设备到云端和云端到设备之间的消息传递，是基于发布/订阅模模式的“轻量级”通信协议。MQTT最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用。

图 2-1 MQTT 协议



数据包组成

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。在MQTT规范中有说明，建议使用**设备侧SDK**进行接入。

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的定义。

- 固定报头：每个 MQTT 控制报文都包含一个固定报头可变报头，用于决定控制报文类型、在PUBLISH中还用于决定最大服务质量（QoS），如：建链、订阅、发布。
- 可变报头：某些 MQTT 控制报文包含一个可变报头部分。它在固定报头和负载之间。可变报头的内容根据报文类型的不同而不同。包含可变报头的报文标识符（Packet Identifier），用于区分同一链路的不同数据包。
- 有效载荷：某些 MQTT 控制报文在报文的最后部分包含一个有效载荷，对于 PUBLISH 来说有效载荷就是应用消息（由用户自己定义）。

说明

MQTT的语法和接口细节，请以**MQTT标准规范**为准。

主要控制报文类型

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

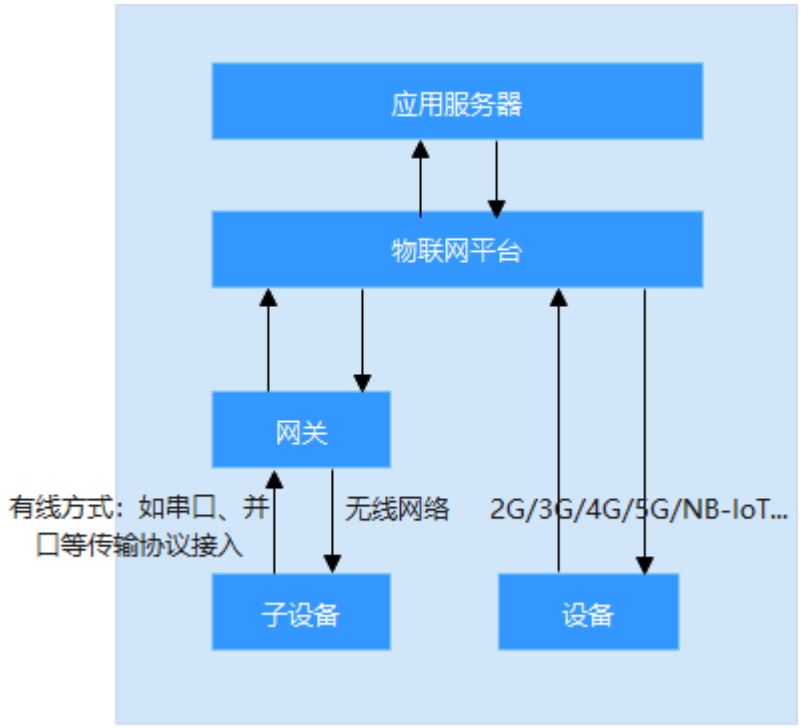
- CONNECT：指客户端请求和服务端连接。有效载荷（Payload）的主要参数，参考**2.4 设备连接鉴权**填写。
- SUBSCRIBE：订阅包括主题筛选器（Topic Filter）和最大服务质量（QoS），一个会话可以进行多个订阅。向平台订阅可以参考：**2.3 Topic定义**。
- PUBLISH：是指从客户端向服务端或者服务端向客户端传输一个应用消息。
  - 可变报头（Variable header）中的主要参数“Topic name”，指服务端或客户端的发布主题。详细请参考**2.3 Topic定义**。
  - 有效载荷（Payload）中的主要参数为完整的数据上报和命令下发的消息内容，一般是一个JSON或字符串对象。
- PINGREQ：指客户端发给服务端的心跳请求。主要用于告知服务端客户端还活着、确认网络连接没有断开。约定的发送周期由协商的Keep Alive决定。

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(华为云平台中具有隐式订阅功能，若通信质量为Qos0，平台的**系统topic**无需订阅。)，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。

2.2 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。

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。

2.3 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
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
	$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
	$oc/devices/{device_id}/sys/gateway/sub_devices/properties/report	设备	平台	网关批量上报属性
	$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

Topic分类	Topic	Publisher(发布者)	Subscriber(订阅者)	用途
	$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	设备事件上报与平台事件下发，可用于：设备网关管理、软固件升级、文件上传/下载、设备时间同步、设备信息上报、设备日志收集、远程配置
	$oc/devices/{device_id}/sys/events/down	平台	设备	

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。
- 除了device_id、request_id其他均为系统字段。

2.4 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

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（device_id）和密钥（secret），生成设备连接鉴权所需的参数（ClientId、Username、Password）。

参数	是否必选	类型	描述
ClientId	是	String	**参数说明：**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。 **取值范围**：长度不超过256
UserName	是	String	**参数说明**：UserName在此处即为设备ID（device_id）。 **取值范围：**长度不超过256
Password	是	String	**参数说明**：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）。 当设备认证类型使用密钥认证接入（SECRET）需填写“Password”，X.509证书认证接入（CERTIFICATES）不需填写“Password”。 **取值范围：**长度不超过256

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。

2.5 设备命令

2.5.1 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。
- 应用通过平台**下发设备命令**时，平台会生成唯一ID（command_id）用于标识该命令，并返回给应用。同时该唯一标识会通过设备命令下行Topic中的requst_id携带给设备。
- 设备无法提前感知该request_id，在订阅该Topic时请使用通配符“#”来替代“request_id={request_id}”即为：$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	• 平台下发时，若为网关子设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。
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	必选	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，其他表示设备执行结果为失败。不带默认为0。
response_name	可选	String	命令的响应名称。
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"
  }
}
```

2.6 设备消息

2.6.1 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格式上报给平台，平台对该消息不进行解析，该消息可以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**的自定义topic。如$oc/devices/{device_id}/user/{自定义}。
- 数据上报的过程中，可以通过在topic后面携带?request_id来指定request_id。比如说消息上报：$oc/devices/{device_id}/sys/messages/up?request_id={request_id}。若不指定，平台会自动生成request_id，用于标识此次请求。

参数说明

消息上报对数据内容不做固定的要求，当使用系统格式进行下发时，参数说明如下：

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
name	可选	String	消息名称，可不填写，做说明用。

字段名	必选/可选	类型	参数描述
id	可选	String	消息的唯一标识，用于区分与查找消息。如不填写系统会自动生成一个消息ID，消息ID不可重复。
content	必选	Object、String	消息内容，可以为base64编码格式。

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

假设设备上报数据内容为"hello!"，请求的样例参考如下：

- 仅发送消息内容：

```
Topic: $oc/devices/{device_id}/sys/messages/up
数据格式：
    hello!
```

- 以系统格式进行上报：

```
Topic: $oc/devices/{device_id}/sys/messages/up
数据格式：
{
    "object_device_id": "{object_device_id}",
    "name":null,
    "id":"aca6a906-c74c-4302-a2ce-b17ba2ce630c",
    "content":"hello!"
}
```

2.6.2 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。在此之上也可以使用平台封装的标准格式，即在应用侧API的**下发设备消息**中payload_format填写为“standard”，或者在设备详情页的**云端下发-消息下发**中下发时选择“按系统格式”。

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

表 2-1 系统格式下发消息字段表

字段名	必选/可选	类型	参数描述
object_device_id	可选	String	• 平台下发时，若为网关子设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。
name	可选	String	消息名称，可不填写，做说明用。
id	可选	String	消息的唯一标识，用于区分与查找消息。如不填写系统会自动生成一个消息ID，消息ID不可重复。
content	必选	String	消息内容，可以为base64编码格式。

示例

系统格式：

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

自定义格式：

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

2.7 设备属性

2.7.1 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

说明

数据上报的过程中，可以通过在topic后面携带?request_id来指定request_id。比如说消息上报：$oc/devices/{device_id}/sys/properties/report?request_id={request_id}。若不指定，平台会自动生成request_id，用于标识此次请求。

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，由创建的**产品模型**确定。
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"
      }
    ]
}
```

2.7.2 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。网关与子设备关系，请查看**网关与子设备**。

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

说明

Topic中device_id为网关设备的设备id。

参数说明

字段名	必选/可选	类型	参数描述
devices	必选	List<DeviceService>	设备数据，是一个字段名为devices的数组，用于存放不同子设备的属性上报。

DeviceService定义表：

字段名	必选/可选	类型	参数描述
device_id	必选	String	此处device_id为网关设备所属的子设备id，用于标记不同子设备的属性上报。
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，由创建的**产品模型**确定。
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。以<属性名:值>的形式存储属性消息。
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"
                }
            ]
        }
```

```
  ]
}
```

2.7.3 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**修改设备属性**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

- 应用**修改设备属性**时，平台会生成唯一ID用于标识该请求。同时该唯一标识会通过平台设置设备属性下行Topic中的requst_id携带给设备。
- 设备无法提前感知该request_id，在订阅该Topic时请使用通配符“#”来替代“request_id={request_id}”即为：$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
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，由创建的**产品模型**确定。
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：
{
    "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"
}
```

2.7.4 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**查询设备属性**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。
- 应用**查询设备属性**时，平台会生成唯一ID用于标识该请求。同时该唯一标识会通过平台查询设备属性下行Topic中的requst_id携带给设备。
- 设备无法提前感知该request_id，在订阅该Topic时请使用通配符“#”来替代“request_id={request_id}”即为：$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
service_id	可选	String	设备的服务ID，由创建的**产品模型**确定。

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，由创建的**产品模型**确定。
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：

```
{
    "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"
        }
    ]
}
```

2.7.5 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: $oc/devices/{device_id}/sys/shadow/get/request_id={request_id}

下行: $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。
- 设备侧上行请求参数中携带唯一标识，该唯一标识会在下行topic中request_id携带给设备以标记唯一请求链。
- 订阅下行Topic时建议使用通配符“#”来替代“request_id={request_id}”即为：$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，由创建的**产品模型**确定。
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
    }
  ]
}
```

2.8 网关与子设备管理

2.8.1 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。网关与子设备关系，请查看**网关与子设备**。

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	该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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。

字段名	必选/可选	类型	参数描述
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
      }
   }]
}
```

2.8.2 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。网关与子设备关系，请查看**网关与子设备**。

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	该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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>	设备信息列表。
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
    }
  }]
}
```

2.8.3 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**2.8.1 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**2.8.2 平台通知网关子设备删除**通知网关设备。网关与子设备关系，请查看**网关与子设备**。

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	该参数目前暂未实装使用，无需填写。
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"。

字段名	必选/可选	类型	参数描述
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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}
   }]
}
```

2.8.4 网关更新子设备状态

功能介绍

网关更新子设备状态，更新结果通过**网关更新子设备状态响应**接口通知网关设备。网关与子设备关系，请查看**网关与子设备**。

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	该参数目前暂未实装使用，无需填写。

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："$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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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
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: $oc/devices/{device_id}/sys/events/up
数据格式：
{
      "object_device_id": "{object_device_id}",
   "services": [{
      "service_id": "$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"
            }
         ]
      }
   }]
}
```

2.8.5 网关更新子设备状态响应

功能介绍

平台将该网关更新子设备状态结果以列表返回，平台收到**网关更新子设备状态**请求后会在30秒内通过此接口返回响应消息。网关与子设备关系，请查看**网关与子设备**。

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	该参数与Topic中的device_id一致。
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："sub_device_update_status"。
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。

字段名	必选/可选	类型	参数描述
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<DeviceStatus>	成功更新的子设备列表详情。
failed_devices	必选	List<Reason>	更新子设备状态失败的原因。

DeviceStatus定义表

字段名	必选/可选	类型	参数描述
device_id	必选	String	设备ID。
status	可选	String	设备更新状态结果。 ● ONLINE：设备在线 ● OFFLINE：设备离线

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	更新失败错误原因码。
error_msg	必选	String	更新失败原因描述。

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": "sub_device_update_status_response",
    "event_time": "20151212T121212Z",
    "event_id": "40cc9ab1-3579-488c-95c6-c18941c99eb4",
    "paras": {
       "successful_devices": [
       {
         "device_id": "c6b39067b0325db34663d3ef421a42f6_subdevice11",
         "status": "ONLINE"
       }
      ],
      "failed_devices": [
       {
         "device_id": "c6b39067b0325db34663d3ef421a42f6_subdevice11",
         "error_code": "XXX",
         "error_msg": "XXXX"
       }
      ]

    }
   }
 ]
}
```

2.8.6 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。网关与子设备关系，请查看**网关与子设备**。

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	该参数目前暂未实装使用，无需填写。
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"。

字段名	必选/可选	类型	参数描述
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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"
         }
       ]
     }
   }
 ]
}
```

2.8.7 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。网关与子设备关系，请查看**网关与子设备**。

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	该参数与Topic中的device_id一致。
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_response"。
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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。
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。

字段名	必选/可选	类型	参数描述
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"
        }
      ]

    }
  }
 ]
}
```

2.8.8 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。网关与子设备关系，请查看**网关与子设备**。

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	该参数目前暂未实装使用，无需填写。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。

字段名	必选/可选	类型	参数描述
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"
        ]
      }
    }
  ]
}
```

2.8.9 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。网关与子设备关系，请查看网关与子设备。

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	该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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。
error_code	必选	String	删除失败错误原因码。

字段名	必选/可选	类型	参数描述
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"
          }
        ]
      }
    }
  ]
}
```

2.9 软固件升级

2.9.1 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	• 平台下发时，若为网关子设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
paras	可选	Object	事件参数JSON对象，下发获取版本信息时无具体字段下发。

paras参数列表

字段名	必选/可选	类型	参数描述
task_id	可选	String	网关模式下，创建软固件升级批量任务的任务ID。
sub_device_count	可选	Integer	网关模式下，同一个软固件升级批量任务中网关设备包含的升级子设备数量。
task_ext_info	可选	Object	创建软固件升级任务中添加的额外扩展信息。

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [
        {
```

```
        "service_id": "$ota",
        "event_type": "version_query",
        "event_time": "20151212T121212Z",
        "paras": {
            "task_id": "65d31bf2581ed33a42a58d76",
            "sub_device_count": 2,
            "task_ext_info": {
                "device_type": "DDC"
            }
        }
    }
  ]
}
```

2.9.2 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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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"。

字段名	必选/可选	类型	参数描述
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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"
        }
    }]
}
```

2.9.3 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	• 平台下发时，若为网关子设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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。
file_name	必选	String	升级包文件名
access_token	可选	String	软固件包url下载地址的临时token。

字段名	必选/可选	类型	参数描述
expires	可选	Integer	access_token的超期时间。
sign	必选	String	软固件包SHA-256值。
custom_info	可选	String	推送给设备的自定义信息。
task_id	可选	String	网关模式下，创建软固件升级批量任务的任务ID。
sub_device_count	可选	Integer	网关模式下，同一个软固件升级批量任务中网关设备包含的升级子设备数量。
task_ext_info	可选	Object	创建软固件升级任务中添加的额外扩展信息。

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)。
file_size	必选	Integer	软固件包文件大小。
file_name	必选	String	升级包文件名
expires	可选	Integer	url的超期时间。
sign	可选	String	上传OBS升级包时输入的SHA256算法计算出的升级包签名值。
custom_info	可选	String	推送给设备的自定义信息。
task_id	可选	String	网关模式下，创建软固件升级批量任务的任务ID。
sub_device_count	可选	Integer	网关模式下，同一个软固件升级批量任务中网关设备包含的升级子设备数量。
task_ext_info	可选	Object	创建软固件升级任务中添加的额外扩展信息。

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://100.93.28.202:8943/iodm/dev/v2.0/upgradefile/applications/******/devices/******/
packages/******",
                "file_size": 81362928,
                "file_name": "upgrade.bin",
                "access_token": "595124473f866b033dfa1f",
                "expires": 86400,
                "sign": "595124473f866b033dfa1f7e831c8c99a12f6143f392dfa996a819010842c99d",
                "custom_info": "This upgrade package adds some new features.",
                "task_id": "65d31bf2581ed33a42a58d76",
                "sub_device_count": 2,
                "task_ext_info": {
                    "device_type": "DDC"
                }
            }
        }
    ]
}
```

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*********",
                "file_size": 81362928,
                "file_name": "upgrade.bin",
                "expires": 3600,
                "sign": "595124473f866b033dfa1f7e831c8c99a12f6143f392dfa996a819010842c99d",
                "custom_info": "This upgrade package adds some new features.",
                "task_id": "65d31bf2581ed33a42a58d76",
                "sub_device_count": 2,
                "task_ext_info": {
                    "device_type": "DDC"
                }
            }
        }
    ]
}
```

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://100.93.28.202:8943/iodm/dev/v2.0/upgradefile/applications/******/devices/******/packages/******
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*********

2.9.4 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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。

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："$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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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：success（处理成功） ● 1：device in use（设备使用中） ● 2：poor signal（信号质量差） ● 3：already the latest version（已经是最新版本） ● 4：low battery（电量不足） ● 5：insufficient storage space（剩余空间不足） ● 6：download timeout（下载超时） ● 7：upgrade package verification failure（升级包校验失败） ● 8：unsupported upgrade package type（升级包类型不支持） ● 9：insufficient memory（内存不足） ● 10：upgrade package installation failure（安装升级包失败） ● 255：internal exception（内部异常）
progress	可选	Integer	设备的升级进度，范围：0到100。
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"
      }
    }]
}
```

2.10 文件上传/下载管理

2.10.1 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

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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。

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
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
            }
         }
    }]
}
```

2.10.2 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

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	• 平台下发时，若为网关设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。

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："$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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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。
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
        }
      }
   }]
}
```

2.10.3 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

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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。

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"
         }
    }]
}
```

2.10.4 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

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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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
         }
      }
  }]
}
```

2.10.5 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

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	• 平台下发时，若为网关子设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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
            }

        }
    }]
}
```

2.10.6 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

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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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。

字段名	必选/可选	类型	参数描述
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"
         }
    }]
}
```

2.11 设备时间同步

2.11.1 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

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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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
         }
    }]
}
```

2.11.2 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

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	• 平台下发时，若为网关子设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。
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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$time_sync",
        "event_type": "time_sync_response",
        "event_time": "20151212T121212Z",
         "paras": {
            "device_send_time": 1582685678789,
            "server_recv_time": 1582685696152,
            "server_send_time": 1582685708109
         }
    }]
}
```

2.12 设备信息上报

2.12.1 设备信息上报

功能描述

设备向平台上报设备信息。

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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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："$sdk_info"。
event_type	必选	String	系统字段，固定为："sdk_info_report"。
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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_v1.0.0等。
sw_version	可选	String	软件版本。
fw_version	可选	String	固件版本。
device_ip	可选	String	设备IP。

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",
            "device_ip": "127.0.0.1"
         }
    }]
}
```

2.13 设备日志收集

2.13.1 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。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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。时间格式为：yyyyMMdd'T'HHmmss'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"
        }
    }]
}
```

2.13.2 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。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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。

字段名	必选/可选	类型	参数描述
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"
        }
    }]
}
```

2.14 远程配置

2.14.1 平台下发配置通知

功能介绍

物联网平台向设备侧下发配置通知。物联网平台为用户提供远程配置功能，用户可以在不中断设备运行的情况下，远程更新设备的系统参数、运行参数等配置信息。例如：Windows的收银台机器，需要通过远程配置修改一些系统参数；车联网中的T-BOX，需要通过远程配置修改设备侧的一些数据上报频率等。

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	• 平台下发时，若为网关子设备，该参数为Topic中设备的子设备Id。 • 平台下发时，若为直连设备，该参数会与Topic中的device_id一致。
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_config"。
event_type	必选	String	系统字段，固定为："config_update"。
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'Z'。
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
config_content	必选	Object	配置的内容。

示例

```
Topic: $oc/devices/{device_id}/sys/events/down
数据格式：
{
    "object_device_id":"{object_device_id}",
    "services":[
        {
            "service_id":"$device_config",
            "event_type":"config_update",
            "event_time":"20151212T121212Z",
            "paras":{
                "config_content":{
                    "config_key1":"device config1",
                    "config_key2":"device config2"
                }
            }
```

```
    }
  ]
}
```

2.14.2 设备上报配置响应

功能介绍

设备上报配置结果响应。创建远程配置任务时，可以配置超时时间（1-30天，不配置默认30天）。在超时时间内，平台每24小时向设备下发一次配置，直到设备上报配置响应。如果设备在超时时间内一直没有上报配置响应，则平台显示该设备配置任务超时失败。

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	• 网关设备上报时，子设备进行上报需填写该参数。object_device_id为Topic中设备的子设备Id，否则将请求失败。 • 直连设备上报时，object_device_id需与Topic中的device_id一致。 • 若为空则默认该参数与Topic中的device_id一致。
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_config"。
event_type	必选	String	系统字段，固定为："config_update_response"。
event_time	可选	String	事件时间。UTC时间，格式：yyyyMMdd'T'HHmmss'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。
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	设备的配置结果，结果码定义如下： ● 0：处理成功。 ● 其他整数：设备自定义异常。
description	可选	String	设备配置执行结果信息，可以返回具体配置失败原因。

示例

```
Topic: $oc/devices/{device_id}/sys/events/up
数据格式：
{
    "object_device_id":"{object_device_id}",
    "services":[
        {
            "service_id":"$device_config",
            "event_type":"config_update_response",
            "event_time":"20151212T121212Z",
            "paras":{
                "result_code":0,
                "description":"update config success"
            }
        }
    ]
}
```

3 设备侧 HTTPS 接口参考

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https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	用于设备将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。
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。
   a. 单击**这里**，填写设备ID和设备密钥，获取时间戳和加密后的password。
   b. 按照**表1 鉴权JSON消息体**编辑您的鉴权JSON消息体，编辑后的消息以**图3-2**为例。

表 3-1 鉴权 JSON 消息体

描述	内容
device_id	您的设备ID
sign_type	建议为0，表示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
timestamp	时间戳，例如2024062602，根据**图1 ClientId生成工具**中的ClientId获取
password	加密后的password，为**图1 ClientId生成工具**中的Password

图 3-1 ClientId 生成工具

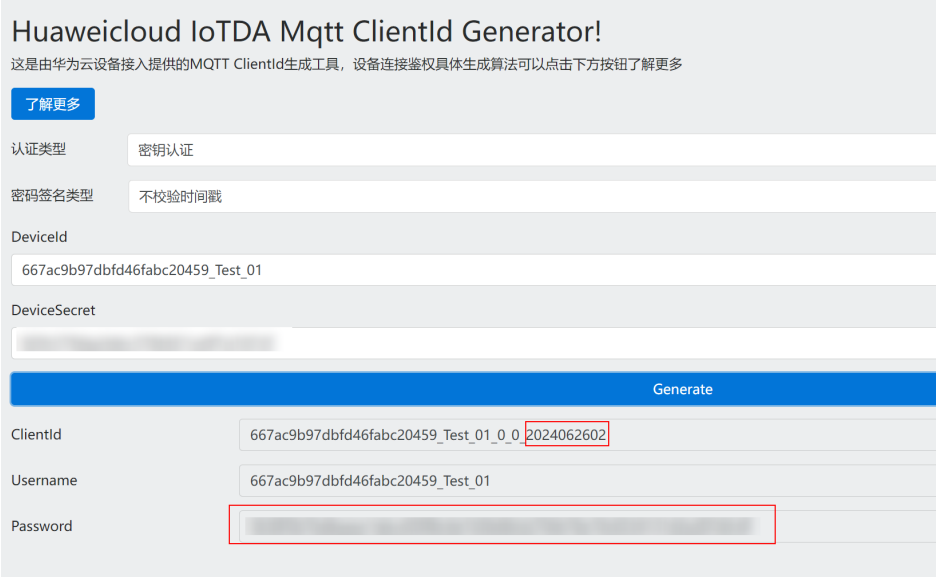
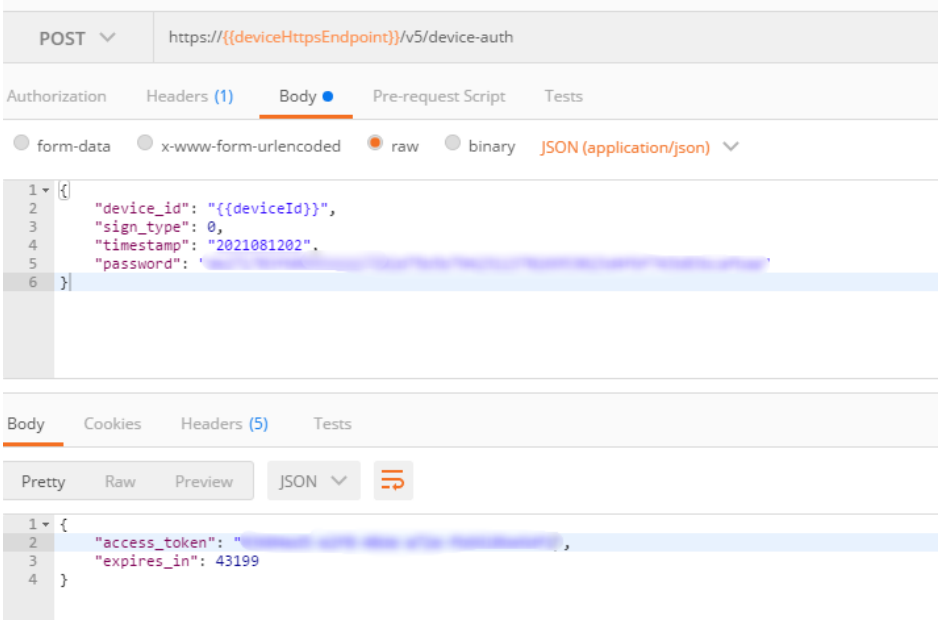


图 3-2 获取设备 access_token



c. 参考**平台对接信息**，获取接入地址，按照**图2 获取设备access_token**拼接成URL后发送，即可获得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 3-3 上报属性

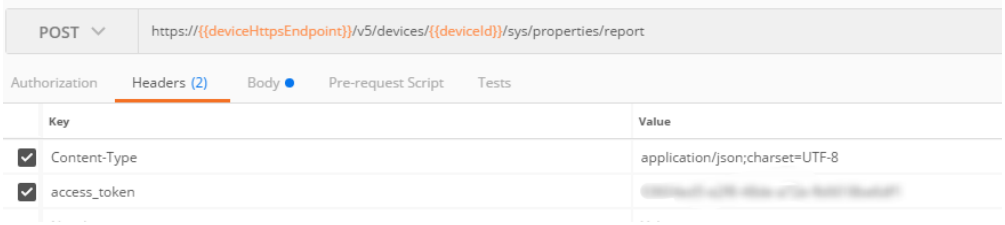
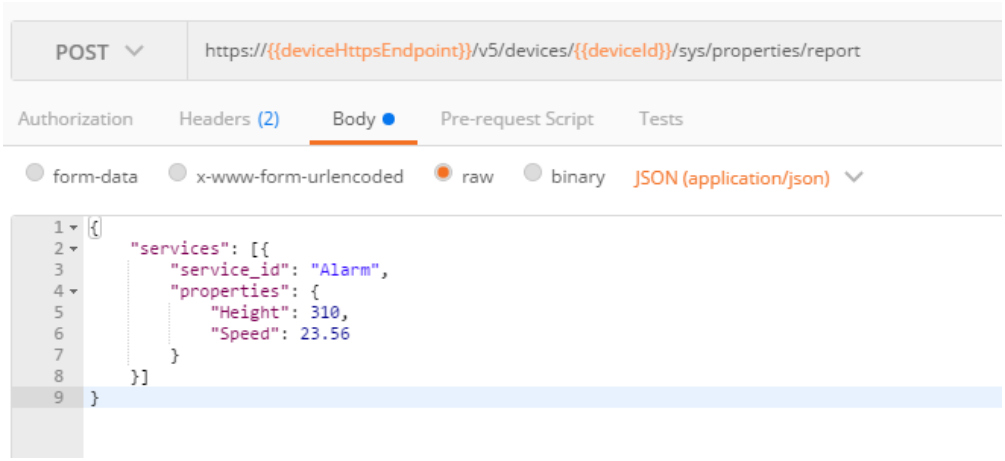


图 3-4 上报属性



https 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

3.2 API 概览

设备鉴权

API	说明
3.3.1 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
3.4.1 设备消息上报	用于设备将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。

设备属性上报

API	说明
3.5.1 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。
3.5.2 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。

3.3 设备鉴权

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，原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:** 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”算法以时间戳为密钥，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" : "******"
}
```

响应示例

Status Code: 200 OK

```
Content-Type: application/json

{
  "access_token" : "******",
  "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

3.4 设备消息上报

3.4.1 设备消息上报

接口说明

用于设备将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。

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:** 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

说明

该接口支持设备将自定义数据通过请求中的body体上报给平台，平台收到该请求后会将body内容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。平台对body中的内容无具体格式限制，小于1MB的数据可以通过该接口携带。

请求示例

```
POST https://{endpoint}/v5/devices/{device_id}/sys/messages/up
Content-Type: application/json
```

```
access_token: ******
{
  "name" : "name",
  "id" : "id",
  "content" : "messageUp"
}
```

响应示例

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

3.5 设备属性上报

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
device_id	是	String	Path	**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 3-2 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'，如2021-08-13T10:10:10.555Z）。 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（格式：yyyyMMdd'T'HHmmss'Z'，如20230523T014506Z）。

请求示例

```
POST https://{endpoint}/v5/devices/{device_id}/sys/properties/report
Content-Type: application/json
access_token: ******

{
  "services" : [ {
    "service_id" : "serviceId",
    "properties" : {
      "Height" : 124,
      "Speed" : 23.24
    },
```

```
    "event_time" : "2021-08-13T10:10:10.555Z"
  } ]
}
```

响应示例

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

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:** 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 3-3 DeviceProperty

名称	必选	类型	说明
device_id	是	String	**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 3-4 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

名称	必选	类型	说明
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'，如2021-08-13T10:10:10.555Z）。 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（格式：yyyyMMdd'T'HHmmss'Z'，如20230523T014506Z）。

请求示例

```
POST https://{endpoint}/v5/devices/{device_id}/sys/gateway/sub-devices/properties/report
Content-Type: application/json
access_token: ******

{
  "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

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

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.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.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

4 设备侧 LwM2M 接口参考

4.1 使用 LwM2M 协议接入

概述

LwM2M（Lightweight M2M，轻量级M2M），由开发移动联盟（OMA）提出，是一种轻量级的、标准通用的物联网设备管理协议，可用于快速部署客户端/服务器模式的物联网业务。LwM2M为物联网设备的管理和应用建立了一套标准，它提供了轻便小巧的安全通信接口及高效的数据模型，以实现M2M设备管理和服务支持。物联网平台支持加密与非加密两种接入设备接入方式，其中加密业务数据交互端口为5684端口，采用DTLS+CoAP协议通道接入，非加密端口为5683，接入协议为CoAP。

说明

LwM2M的语法和接口细节，请以此**标准规范**为准。

物联网平台支持协议规定的plain text, opaque, Core Link ,TLV , JSON编码格式。在多字段操作时（比如写多个资源），默认用TLV格式。

使用限制

表 4-1 使用限制

描述	限制
支持的LwM2M协议版本	1.1
支持的DTLS版本	DTLS 1.2

描述	限制
支持的加密算法套件	TLS_PSK_WITH_AES_128_CCM_8，TLS_PSK_WITH_AES_128_CBC_SHA256
支持的body体最大长度	1KB
接口规格说明	请参考**产品规格说明**。

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-> CoAP (5683)| CoAPS (5684)”对应的Endpoint，端口为5683（非加密接入方式）或者5684（加密接入方式）。

4.2 API 概览

物联网平台支持的 LwM2M 对象资源

表 4-2

资源路径	对象名	资源名	对应平台的功能
/rd?ep={nodeId}	Device	Register	设备连接
/3/0/3	Device	Firmware Version	查询固件版本号
/4/0/0	Connectivity monitoring	Network Bearer	识别设备接入网络类型
/4/0	Connectivity monitoring	Network Signal Strength	查询信号强度
/4/0/8	Connectivity monitoring	Cell ID	小区ID
/5/0/1	Firmware Update	Package URI	下发固件版本下载链接
/5/0/2	Firmware Update	Update	通知固件升级
/5/0/3	Firmware Update	State	固件升级状态通知
/5/0/5	Firmware Update	Update Result	查询固件升级结果
/19/0/0	BinaryAppDataContainer	Data	上行业务消息传输

资源路径	对象名	资源名	对应平台的功能
/19/1/0	BinaryAppDataContainer	Data	下行业务数据传输

说明

BinaryAppDataContainer对象：

为简化设备厂商定义并使用LwM2M对象的复杂度，华为定义了19对象，由芯片或模组实现，设备只需要调用AT接口或函数接口来实现业务数据的收发，不需要关心LWM2M协议，参考此**规范**。

设备鉴权

API	说明
4.3 设备鉴权	设备向物联网平台注册，物联网平台进行设备的身份认证。物联网平台向设备订阅资源，设备获取到物联网平台下发订阅时的token，调用属性上报等其他接口时，都需要携带token信息。

设备属性上报

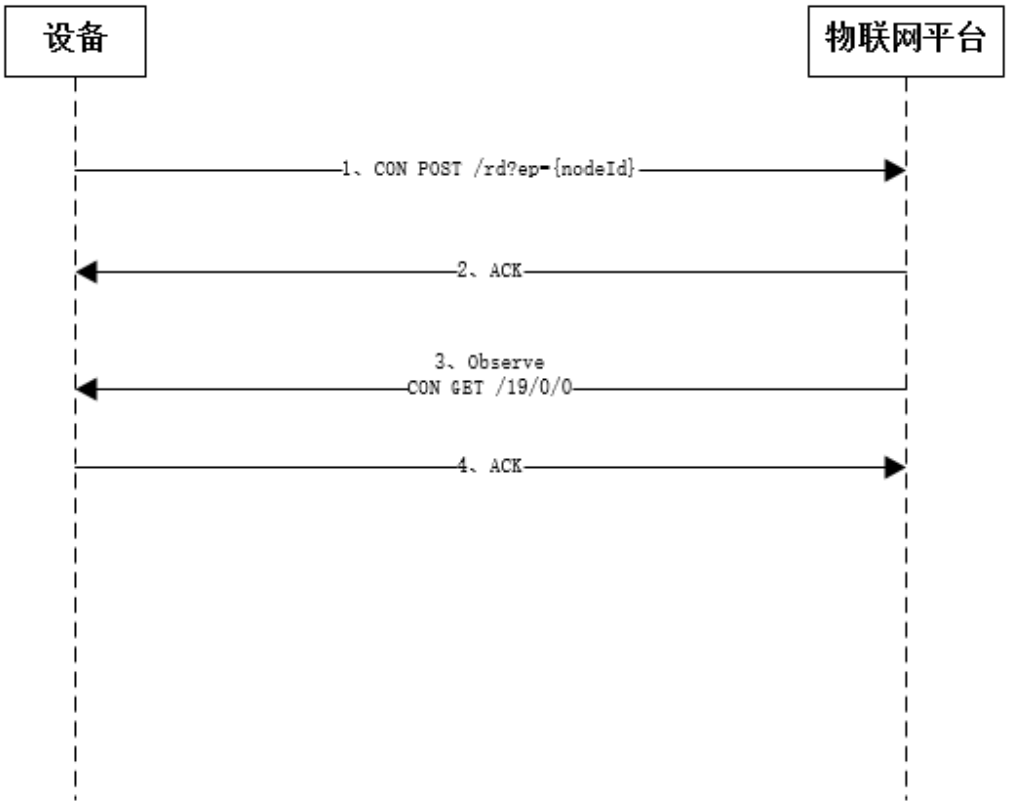
API	说明
4.4 设备属性上报	设备向平台上报数据，设备携带平台下发/19/0/0订阅时的token进行属性上报。

设备命令下发

API	说明
4.5 设备命令下发	物联网平台向设备下发命令，物联网平台通过LwM2M的/19/1/0资源对象将命令打包到LwM2M write消息的payload里下发给设备。

4.3 设备鉴权

流程说明



LwM2M 对象资源

设备向物联网平台注册，物联网平台进行设备的身份认证。

Operation	CoAP Method	URI	Success	Failure
Register	POST	/rd?ep={nodeId}	2.01 Created	4.00 Bad Request, 4.03 Forbidden, 4.12 Precondition Failed

物联网平台向设备订阅资源，设备获取到物联网平台下发订阅时的token，调用属性上报等其他接口时，都需要携带token信息。

Operation	CoAP Method	URI	Success	Failure
Observe	GET with Observe option = 0	/19/0/0	2.04 Content with Observe option	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed

请求参数

名称	必选	类型	位置	说明
nodeId	是	String	Uri-Query	设备标识码，用于唯一标识一个设备。在物联网平台注册设备时直接指定。

Register 请求示例

```
CON-POST    MID=25995, Token=514078a73366, OptionSet={"Uri-Path":"rd", "Content-Format":"application/octet-stream", "Uri-Query":["ep=test"]}
```

Register 响应示例

```
ACK-2.01    MID=25995, Token=514078a73366, OptionSet={"Location-Path":["rd","test"]}, no payload：
```

Observe 请求示例

```
CON-GET    MID=48590, Token=2cb6a673cba24c04, OptionSet={"Observe":0, "Uri-Path":["19","0","0"], "Accept":"application/octet-stream"}, no payload
```

Observe 响应示例

```
ACK-2.04    MID=48590, Token=2cb6a673cba24c04, OptionSet={"Observe":0, "Content-Format":"application/octet-stream"}
```

4.4 设备属性上报

LwM2M 对象资源

设备向物联网平台上报数据，设备携带物联网平台下发/19/0/0订阅时的token进行属性上报。

Operation	CoAP Method	URI	Success	Failure
Notify	Asynchronous Response	不涉及	2.05 Content with {value}	不涉及

说明

设备通过LwM2M协议上报的数据都是二进制报文数据，一般都需要通过编解码插件进行解析，参考**编解码插件开发**。

请求参数

参数名	参数位置	可选/必选	说明
value	Payload	必选	设备上报的数据内容

请求示例

```
假设value为c4 0d 5a 6e 96 0b c3 0e 2b 30 37，则上报样例如下：
NON-2.05    MID=48590, Token=2cb6a673cba24c04, OptionSet={"Observe":22, "Content-Format":"application/octet-stream"}, c4 0d 5a 6e 96 0b c3 0e 2b 30 37
```

4.5 设备命令下发

LwM2M 对象资源

物联网平台向设备**1.4.4.2.1 下发异步设备命令**，物联网平台通过LwM2M的/19/1/0资源对象将命令打包到LwM2M write消息的payload里下发给设备。

Operation	CoAP Method	URI	Success	Failure
Write	PUT	/19/1/0	2.04 Changed	4.00 Bad Request, 4.04 Not Found, 4.01 Unauthorized, 4.05 Method Not Allowed, 4.06 Not Acceptable

说明

1、使用LwM2M协议接入的设备，物联网平台只支持**1.4.4.2.1 下发异步设备命令**。

2、物联网平台接收到命令之后会调用用户的在物联网平台上传的编解码插件进行编码后下发给设备，参考**编解码插件开发**。

请求示例

假设最终通过编解码插件编码后的二进制value为c4 0d 5a 6e 96 0b c3 0e 2b 30 37，则下发样例如下：
CON-PUT MID=48587, Token=2d82e7f54b, OptionSet={"Uri-Path":["19","1","0"], "Content-Format":"application/octet-stream"}, c4 0d 5a 6e 96 0b c3 0e 2b 30 37

响应示例

ACK-2.04 MID=48587, Token=2d82e7f54b, OptionSet={"Content-Format":"application/octet-stream"}, no payload

5 模组 AT 指令参考

5.1 AT 指令列表

经过兼容性认证的模组，在AT命令以及格式规范上，基本和华为通用要求一致，部分模组厂家受限于自己的AT通道，实现稍有不同，可以参考模组厂家的特殊说明。

AT指令	说明
AT+HMVER	获取华为SDK的版本信息
AT+HMCON	设置MQTT协议连接参数
AT+HMDIS	关闭和华为IoT平台连接
AT+HMPUB	发送MQTT数据到指定TOPIC
+HMREC	模组接收到的数据通过该方式传递给外部MCU
+HMSTS	模组连接或者断开的状态主动传递给外部MCU。
AT+HMSUB	订阅自定义主题

AT指令	说明
AT+HMUNS	取消订阅自定义主题
AT+HMPKS	用于设置服务器或者客户端证书

5.2 AT+HMVER

该指令用于获取华为SDK的版本信息。

指令	参数	参数缺省处理	AT响应结果	使用示例
AT +HMVER	-	-	+HMVER:vx.x.x AT TIME ON DATE	AT+HMVER

5.3 AT+HMCON

该指令用于设置MQTT协议连接参数。

指令	参数	参数缺省处理	AT响应结果	使用示例
AT +HMCON=bsmode,lifetime,serverip,serverport,deviceid,passwd,codec	bsmode：0表示非bs模式，1表示bs模式。	默认值：0	-	AT +HMCON=0,10,"iot-mqtts.cn-north-4.myhuaweicloud.com","8883","testID","testPasswd",0
	lifetime：mqtt保持心跳时间，建议大于等于30。	默认值：300	+HMCON OK（连接成功）	
	serverip：设备接入服务器地址或者BS服务器地址。	默认值：BS服务器地址	+HMCON ERR：code（连接失败，code表示失败原因码）	
	serverport：设备接入服务器端口或者BS服务器端口。	默认值：MQTTS端口8883	-	
	id：设备ID，最大长度256。	注册设备后获取	-	
	pwd：设备密钥，最大长度256。	注册设备后获取	-	
	codec：payload编码方式，0：原始编码，1：十六进制编码。	默认值：0	-	

5.4 AT+HMDIS

该指令用于关闭和华为IoT平台连接。

指令	参数	参数缺省处理	AT响应结果	使用示例
AT+HMDIS	-	-	+HMDIS OK（断开成功） +HMDIS ERR：code（断开操作失败，code表示失败原因）	AT+HMDIS

5.5 AT+HMPUB

该指令用于发送MQTT数据到指定TOPIC。

指令	参数	参数缺省处理	AT响应结果	使用示例
AT+HMPUB=qos,topic,payload_len,payload	qos：mqtt的qos（0，1，2）	默认值：0	+HMPUB OK（发布成功）	hexstring模式下，发送数据的byte流必须转换为ascii码的hexstring。 如发送字节0x12 0x34两字节到"test"：AT+HMPUB=0,"test",2,1234
	topic：指定的topic	默认为上报属性的topic，建议填写	+HMPUB ERR：code（发布失败，code表示失败原因）	
	len：待发送数据长度，单位为字节	-	-	
	payload：转换为ascii码的数据，最大长度1024字节	最大支持长度2KB	-	

5.6 +HMREC

该指令用于将模组接收到的数据通过该方式传递给外部MCU。

指令	参数	参数缺省处理	AT响应结果	使用示例
+HMREC=topic, payload_len, payload	topic：指定的topic	-	-	+HMREC="$oc/devices/my_deviceid/user/my_subtopic",2,0102(hexmode)
	len：接收到的数据长度，单位为字节	-	-	
	payload：转换为ascii码的数据	-	-	

5.7 +HMSTS

该指令用于将模组连接或者断开的状态主动传递给外部MCU。

指令	参数	参数缺省处理	AT响应结果	使用示例
+HMSTS: status	status：当前的状态。0表示已连接，1表示已断开。 如果在通信过程中因为网络原因断开，模组会尝试重连并订阅已经订阅的主题。	-	-	+HMSTS:0 表示和IoT平台已经连接上 +HMSTS:1 表示网络原因和IoT平台断开

5.8 AT+HMSUB

该指令用于订阅自定义主题。

指令	参数	参数缺省处理	AT响应结果	使用示例
AT +HMSUB=qos, topic	qos：topic的qos	默认值：0	+HMSUB OK when success	AT+HMSUB=0,"$oc/devices/my_deviceid/user/my_subtopic"
	topic：自定义主题	自定义	+HMSUB ERR:code code:reference to en_oc_mqtt_err_code_t defines	

5.9 AT+HMUNS

该指令用于取消订阅自定义主题。

指令	参数	参数缺省处理	AT响应结果	使用示例
AT+HMUNS=topic	topic：自定义主题	自定义	+HMUNS OK when success +HMUNS ERR:code code:reference to en_oc_mqtt_err_code_t defines	AT+HMUNS="$oc/devices/my_deviceid/user/my_subtopic"

5.10 AT+HMPKS

该指令用于设置服务器或者客户端证书。

指令	参数	参数缺省处理	AT响应结果	使用示例
AT+HMPKS =type,para1,[para2]	type=0，表示平台证书，证书在para1中。 type=1，表示设备公钥证书，证书在para1中。 type=2，表示设备私钥证书，证书在para1中，密码在para2中。 para1用于存放证书，当为空时表示清除证书。 para2用于存放私钥证书的密码，只有当设置私钥证书时有效，并且证书以字符串格式传输PEM。	自定义	+HMPKS OK when success +HMPKS ERR when failed	AT+HMPKS =0,"SERVER_CA" 证书中不要包含“"”

6 修订记录

发布日期	修订记录
2024-06-25	第四十一次正式发布 新增 • 1.4.24.1 创建设备策略 • 1.4.24.2 查询设备策略列表 • 1.4.24.3 删除设备策略 • 1.4.24.4 查询设备策略详情 • 1.4.24.5 更新设备策略信息 • 1.4.24.6 绑定设备策略 • 1.4.24.7 解绑设备策略 • 1.4.24.8 查询设备策略绑定的目标列表 • 1.4.25.1 创建预调配模板 • 1.4.25.2 查询预调配模板列表 • 1.4.25.3 删除预调配模板 • 1.4.25.4 查询预调配模板详情 • 1.4.25.5 更新指定id的预调配模板信息 • 1.4.26.1 创建自定义鉴权 • 1.4.26.2 查询自定义鉴权列表 • 1.4.26.3 删除自定义鉴权 • 1.4.26.4 查询自定义鉴权详情 • 1.4.26.5 更新指定id的自定义鉴权

发布日期	修订记录
2024-05-30	第四十次正式发布 新增 • **1.4.23.1 创建网桥** • **1.4.23.2 查询网桥列表** • **1.4.23.3 删除网桥** • **1.4.23.4 重置网桥密钥** • **1.4.16.4 更新CA证书**
2024-04-26	第三十九次正式发布 新增 • **1.4.22.1 创建设备代理** • **1.4.22.2 查询设备代理列表** • **1.4.22.3 查询设备代理详情** • **1.4.22.4 修改设备代理** • **1.4.22.5 删除设备代理**
2024-04-11	第三十八次正式发布 新增 • **新建数据流转积压策略** • **查询数据流转积压策略列表** • **修改数据流转积压策略** • **查询数据流转积压策略** • **删除数据流转积压策略** • **新建数据流转流控策略** • **查询数据流转流控策略列表** • **修改数据流转流控策略** • **查询数据流转流控策略** • **删除数据流转流控策略**
2024-03-29	第三十七次正式发布 新增 • **更新资源空间** 修改 • **创建批量任务**、**查询批量任务列表**、**查询批量任务**接口，新增task_mode参数用于软固件升级任务支持网关模式，task_ext_info参数用于支持添加批量任务的额外扩展信息。

发布日期	修订记录
2024-03-13	第三十六次正式发布 修改 **上传设备CA证书**、**获取设备CA证书列表**接口新增返回参数，provision_enable参数用于确认该CA证书是否开启自注册能力，template_id参数为开启自注册能力后关联的模板ID。
2024-02-27	第三十五次正式发布 修改 **创建规则**、**查询规则列表**、**修改规则**、**查询规则**接口，调整ActionSmnForwarding结构中的message_content参数为非必选参数，调整message_title参数的取值范围，新增message_template_name参数用于提供选择SMN模板的能力。
2024-01-05	第三十四次正式发布 修改 • **生成接入凭证**：新增参数force_disconnect，用于生成接入凭证时供用户选择是否强制断开北向MQTT/AMQP的连接。
2023-12-06	第三十三次正式发布 修改 • **重置设备密钥**：新增参数secret_type，用于设置/更新设备辅密钥。 • **重置设备指纹**：新增参数fingerprint_type，用于设置/更新设备辅指纹。 • **查询设备**、**1.4.2.1 创建设备**、**1.4.2.4 修改设备**：新增响应参数secondary_secret、secondary_fingerprint，用于查询出设备的辅指纹/辅密钥。
2023-11-15	第三十二次正式发布 修改**创建规则动作**、**修改规则动作**、**查询规则动作列表**、**查询规则动作**接口。 • 调整dms_kafka_forwarding结构中的mechanism参数枚举值，新增security_protocol参数用以完整的支持kafka的安全认证能力。 • 新增http_forwarding结构中的signature_enable、token参数，用以提供HTTP推送多样化的安全认证能力。
2023-10-23	第三十一次正式发布 修改 **查询产品列表**，新增product_name入参，支持产品名称检索能力。

发布日期	修订记录
2023-07-21	第三十次正式发布 新增 • **删除批量任务** • **创建设备隧道** • **查询设备所有隧道** • **查询设备隧道** • **关闭设备隧道** • **删除设备隧道**
2023-06-17	第二十九次正式发布 新增 • **重试批量任务** • **停止批量任务** 修改 **设备同步命令**、**修改设备属性**接口，增加响应参数error_code、error_msg，用以支持返回同步操作时设备的异常行为，如超时未响应。
2023-05-13	第二十八次正式发布 修改 **添加设备组**、**修改设备组**、**查询设备组列表**、**查询设备组**接口，新增group_type、dynamic_group_rule参数，用以支持动态设备组能力。
2023-04-19	第二十七次正式发布 新增 **下发广播消息**接口。 修改 **创建批量任务**、**查询批量任务列表**、**查询批量任务**，修改task_type参数，增加支持"updateDevices"枚举值，用以支持批量更新设备任务。
2023-03-29	第二十六次正式发布 修改 **创建规则**、**修改规则**、**查询规则列表**、**查询规则**接口，新增device_linkage_status_condition参数，用于支持设备状态变化的联动规则条件。

发布日期	修订记录
2023-02-24	第二十五次正式发布 新增 • **创建OTA升级包** • **查询OTA升级包列表** • **查询OTA升级包详情** • **删除OTA升级包** 修改 修改**查询设备**接口，新增响应参数connection_status_update_time、active_time参数用以查询设备的连接状态变更时间和激活时间。
2023-02-07	第二十四次正式发布 修改 **创建规则动作**、**修改规则动作**、**查询规则动作列表**、**查询规则动作**接口，新增dms_rocketmq_forwarding、mrs_kafka_forwarding、roma_forwarding、influxdb_forwarding参数用以支持对rocketmq、kafka、roma、influxdb的消息流转能力。
2022-12-14	第二十三次正式发布 新增 **灵活搜索设备列表**接口。 修改 修改**创建规则**、**修改规则**、**查询规则列表**、**查询规则**接口，新增rule_type参数的枚举值"DEVICE_SIDE"，用以支持设备端测规则联动的能力。
2022-09-20	第二十二次正式发布 修改**下发设备消息**、**查询设备消息**接口，新增properties参数用于支持MQTT5.0特性能力。
2022-03-21	第二十一次正式发布 新增 • **重置设备指纹**接口。
2021-06-18	第二十次正式发布 修改 修改应用侧API接口参数的类型： • 去除长度描述。 • 对象参数增加“Object”。
2021-05-28	第十九次正式发布 修改 修改应用侧API接口请求参数描述，增加参数说明和取值范围。

发布日期	修订记录
2021-02-08	第十八次正式发布 新增 **设备添加通知**、**设备更新通知**接口中，增加device_sdk_version字段类型。
2020-12-21	第十七次正式发布 修改 **1.4.2.3 查询设备**、**1.4.2.4 修改设备**接口中，补充secret字段和timeout字段说明。
2020-12-07	第十六次正式发布 新增 **1.4.8.2 查询规则条件列表**、**1.4.8.1 创建规则触发条件**、**1.4.8.3 查询规则条件**、**1.4.8.4 修改规则触发条件**接口中，增加resource字段类型。
2020-11-03	第十五次正式发布 新增 **1.4.8.7 查询规则动作列表**、**1.4.8.6 创建规则动作**、**1.4.8.8 查询规则动作**、**1.4.8.9 修改规则动作**、**1.4.8.10 删除规则动作**接口中，增加规则动作的类型。
2020-10-30	第十四次正式发布 修改 原规则管理接口删除了响应参数DATA_FORWARDING、EDGE，仅支持设备联动规则，并更名为**1.4.10 设备联动规则**。如需数据转发功能，可使用接口**1.4.8 数据流转规则管理**。
2020-09-17	第十三次正式发布 新增 • **2.8.6 网关新增子设备请求** • **2.8.7 网关新增子设备请求响应** • **2.8.8 网关删除子设备请求** • **2.8.9 网关删除子设备请求响应**
2020-09-11	第十二次正式发布 新增 • **1.4.6.2 查询AMQP列表** • **1.4.6.1 创建AMQP队列** • **1.4.6.3 查询单个AMQP队列** • **1.4.6.4 删除AMQP队列** • **1.4.7.1 生成接入凭证** • **1.4.9 流转数据** • **1.4.8 数据流转规则管理**

发布日期	修订记录
2020-08-21	第十一次正式发布 新增 • **1.4.4.2.1 下发异步设备命令** • **1.4.4.2.2 查询指定id的命令** • 补充错误码中文描述和处理建议。
2020-08-18	第十次正式发布 修改 **2.10.1 设备上报获取文件上传URL请求**、**2.10.2 平台下发文件上传临时URL**、**2.10.4 设备上报获取文件下载URL请求**、**2.10.5 平台下发文件下载临时URL**接口中，增加扩展参数file_attribute。 新增 **2.7.1 设备属性上报**和**2.7.2 网关批量设备属性上报**接口中，event_time字段，时间格式支持毫秒级别。
2020-08-10	第九次正式发布 新增 **1.4.15.1 创建批量任务**task_type参数中，新增freezeDevices和unfreezeDevices字段，支持批量冻结设备、批量解冻设备。
2020-07-17	第八次正式发布 新增 • **1.4.14.1 查询资源空间列表** • **1.4.14.2 创建资源空间** • **1.4.14.3 查询资源空间** • **1.4.14.4 删除资源空间**
2020-06-28	第六次正式发布 新增 • 在**1.4.2.7 冻结设备**和**1.4.2.8 解冻设备**中，补充IOTDA.014038和IOTDA.014039错误码，以及错误码描述信息。 • 补充IOTDA.000022错误码描述信息。 • 补充文档修订记录。
2020-06-24	第五次正式发布 新增 • **1.4.2.7 冻结设备** • **1.4.2.8 解冻设备** • **1.4.15.7.2 查询批量任务文件列表** • **1.4.15.7.1 上传批量任务文件** • **1.4.15.7.3 删除批量任务文件** • **5 模组AT指令参考**

发布日期	修订记录
2020-06-08	第四次正式发布 修改 • **1.4.15 批量任务**，删除freezeDevices和unfreezeDevices参数。 • 新增通过**API Explorer**中直接运行调试该接口。 • 修改X-Auth-Token参数的描述信息。 • **1.4.2.1 创建设备**，修改app_id参数的描述信息。
2020-06-02	第三次正式发布 新增 • **2.10.1 设备上报获取文件上传URL请求** • **2.10.2 平台下发文件上传临时URL** • **2.10.3 设备上报文件上传结果** • **2.10.4 设备上报获取文件下载URL请求** • **2.10.5 平台下发文件下载临时URL** • **2.10.6 设备上报文件下载结果**
2020-05-26	第二次正式发布 修改 **2.7.5 设备侧获取平台的设备影子数据**，修改request_id的描述。
2020-03-03	第一次正式发布 **1 应用侧API参考** **2 设备侧MQTT/MQTTS接口参考**